
VNIVERSITAT DE VALÈNCIA

Facultat de Ciències Matemàtiques

Departament d'Estadística i Investigació Operativa

Programa de doctorat en Estadística i Optimització

Doctoral Thesis

# Contributions to Close-Enough Arc Routing Problems

*Author*

Miguel Reula Martín

*Supervisors*

Dr. Ángel Corberán Salvador
Dr. Isaac Plana Andani
Dr. José María Sanchis Llopis

May 2021

*Some people want it to happen,*
*some wish it would happen,*
*others make it happen.*

- Michael Jordan

# Acknowledgements

This doctoral thesis has been carried out under FPI Grant BES-2016-077473, awarded by the Spanish Ministerio de Economía y Competitividad (MINECO) and under projects MTM2015-68097-P and PGC2018-099428-B-I00, which provided support to participate in scientific conferences, meetings, and courses. Many people have contributed to the fulfilment of this thesis and I would like to thank them for their direct or indirect participation.

First, I would like to express my gratitude to my thesis supervisors, Ángel, José María and Isaac, for their dedication during the course of this thesis and for giving me the opportunity to be part of the great scientific and personal team they form. Thank you very much Ángel, for being the best role model. I hope you continue to share your wisdom in the new stage you are starting, because you still have a lot to teach and because all of us who want to dedicate ourselves to the field of Operational Research have a lot to learn from you. I also want to thank Nicola Bianchessi, supervisor of the research visit I made while working on this thesis at the Università degli Studi di Milano (Italy), for his kindness and hospitality. I am very grateful for his contribution to this thesis, for all I learned working side by side with him during my stay, as well as for the effort he made, despite the difficulties, to complete the project once my stay finished. It has been a pleasure to meet and work with you.

To all the members of the Department of Statistics and Operational Research, thank you for the training, support and help when needed. It has been a pleasure to be part of this amazing team. In particular, thanks to all my fellow PhD students who have become friends. We have shared moments that will always remain in my heart. All of them have contributed to make the work more enjoyable and to motivate me every day.

I would especially like to thank my family for their unconditional support. Even though life as a mathematician did not get off to a good start, they have never stopped supporting me and have kept me strong in the worst moments. To my parents for instilling in me that tireless desire to constantly work and try to improve in everything I do. To my second parents, my uncles, who although they are not here to celebrate this achievement, they are and will always be in my memory. To my sister for her constant motivation when I needed it most. To Javi for his endless advice, and for being the father of a little mini-me, Iago.

I do not know if I should also thank my friends for this thesis, as it has often been very difficult for me to decide between working or their plans. What I do know is that Cabanes and them have been my best medicine and those hours of bar, friends, sports, beer, and laughter, made me happy and motivated me to continue working. Finally, thanks to my best friend, my best work partner and, above all, my best life partner. To you, Consu, for having trusted me more than I trust myself. Without you this would not have been possible. Thank you for understanding each of my moods and for giving me the best advice in each of my disappointments.

# Contents

**Acknowledgements** **v**

**List of Figures** **x**

**List of Tables** **xii**

**List of Problems** **xv**

**List of Acronyms** **xvii**

**1 Introduction** **1**
1.1 Motivation: Transportation and Routing Problems . . . . . 1
1.2 Scope of this thesis: CEARP . . . . . 4
1.3 Outline . . . . . 6

**2 Preliminaries** **9**
2.1 Graph Theory . . . . . 9
2.2 Linear and Integer Programming . . . . . 12
2.3 Polyhedral theory and polyhedral combinatorics . . . . . 14

**3 Routing Problems** **19**
3.1 Node Routing Problems . . . . . 21
3.2 Arc Routing Problems . . . . . 22
3.3 General Routing Problems . . . . . 26

**4 Close-Enough Routing Problems** **29**
4.1 Introduction . . . . . 29
4.2 Close-Enough Traveling Salesman Problems . . . . . 32
4.3 Close-Enough Arc Routing Problems . . . . . 33
4.3.1 Literature review . . . . . 33
4.3.2 CEARP definition and formulation . . . . . 34

4.3.3 CEARP variants . . . 38
Profitable CEARP . . . 38
Distance-Constrained CEARP . . . 38
Min-Max CEARP . . . 39
Other related problems . . . 40

**5 The Profitable CEARP 41**
5.1 Problem definition and formulation . . . 43
5.2 The PCEARP polyhedron . . . 44
5.2.1 Facet-inducing inequalities obtained from the formulation . . . 45
5.2.2 Additional valid inequalities . . . 50
Parity inequalities . . . 51
K-C inequalities . . . 53
5.3 A heuristic for the PCEARP . . . 57
5.4 A branch-and-cut algorithm for the PCEARP . . . 59
5.4.1 Separation algorithms . . . 59
Connectivity inequalities . . . 60
Parity inequalities . . . 60
K-C inequalities . . . 61
5.4.2 Cutting-plane algorithm . . . 62
5.4.3 Lower bound heuristic . . . 62
5.5 Computational experiments . . . 63
5.5.1 Instances . . . 63
5.5.2 Computational results . . . 64
5.6 Conclusions . . . 67

**6 A Matheuristic for the DC-CEARP 69**
6.1 Introduction . . . 69
6.2 Problem definition and notation . . . 70
6.3 Matheuristic . . . 71
6.3.1 Constructive algorithm . . . 71
Initialization procedures . . . 72
Route completion procedures . . . 73
6.3.2 Local search . . . 74
2-Exchange . . . 74
Destroy and repair . . . 74
6.3.3 Route optimization . . . 75
6.3.4 Overall algorithm . . . 75
6.4 Computational experiments . . . 77

6.4.1 Instances . . . 77
6.4.2 Computational results . . . 77
6.5 Conclusions . . . 82

**7 On the Distance-Constrained CEARP** **83**
7.1 Introduction . . . 83
7.2 Problem definition and formulation . . . 84
7.3 Valid inequalities . . . 86
7.3.1 More connectivity inequalities . . . 86
7.3.2 Parity inequalities . . . 87
7.3.3 K-C inequalities . . . 89
7.3.4 K-$C_{02}$ inequalities . . . 95
7.3.5 Path-Bridge inequalities . . . 98
7.3.6 Max-distance constraints . . . 103
7.3.7 Symmetry breaking inequalities . . . 104
7.4 The Branch-and-Cut Algorithm . . . 105
7.4.1 Separation algorithms . . . 105
Connectivity inequalities . . . 105
Parity inequalities . . . 106
K-C, K-$C_{02}$ and Path-Bridge inequalities . . . 108
Max-distance inequalities . . . 109
Inequalities and separation procedures: a summary . . . 109
7.4.2 Comparison of separation strategies and cutting-plane algorithms 112
7.5 Computational experiments . . . 117
7.5.1 Instances . . . 118
7.5.2 Computational Results . . . 119
7.6 Conclusions . . . 124

**8 On the Min-Max CEARP** **125**
8.1 Introduction . . . 125
8.2 Problem definition and formulation . . . 126
8.2.1 Arc-based formulation . . . 127
Parity inequalities . . . 128
Max-distance constraints . . . 129
8.2.2 Route-based formulation . . . 130
8.3 Branch-and-cut algorithm . . . 132
8.3.1 Separation algorithms . . . 132
8.3.2 Initial relaxation and cutting-plane algorithm . . . 133
8.4 Branch-and-price algorithm . . . 134

8.4.1 Column generation . . . 134
Pricing problem modeling . . . 135
A branch-and-cut algorithm for the pricing problem . . . 136
Solution of the PPs . . . 140
Heuristic column generation . . . 140
Restricted master heuristic . . . 141
Overall algorithm overview . . . 142
8.4.2 Branching rules . . . 144
8.5 Primal bound heuristic . . . 146
8.6 Computational experiments . . . 147
8.6.1 Instances . . . 147
8.6.2 Computational Results . . . 148
8.7 Conclusions . . . 154

**9 Conclusions and future work** **155**
9.1 Contributions . . . 155
9.2 Future lines of research . . . 158

**A Resumen Extendido** **159**

**Bibliography** **169**

# List of Figures

1.1 Transport infrastructure. . . . . 2
1.2 The Seven Bridges of Königsberg. . . . . 4
1.3 Automatic Meter Reading. . . . . 5

3.1 Applications of routing problems . . . . . 20
3.2 Chinese Postman Problem (CPP) . . . . . 24
3.3 Rural Postman Problem (RPP) . . . . . 25
3.4 Capacitated Arc Routing Problem (CAPP) . . . . . 26

4.1 CERP application in meter reading . . . . . 30
4.2 Application CERP in Inventory Review . . . . . 31
4.3 Application CERP in surveillance tasks . . . . . 31
4.4 Close-Enough Arc Routing Problem (CEARP) . . . . . 35

5.1 Matrices appearing in the proof of Theorem 16 . . . . . 48
5.2 Matrix appearing in the proof of Theorem 17 . . . . . 51
5.3 Parity inequalities (5.6) . . . . . 52
5.4 A K-C structure. . . . . 54
5.5 Structure of a 3-C inequality . . . . . 55

7.1 Structure of the parity inequalities for the DC-CEARP . . . . . 87
7.2 Standard K-C basic structure. . . . . 89
7.3 Standard disaggregate K-C inequality for the DC-CEARP. . . . . 90
7.4 A fractional solution for vehicle $k$ not cut off by a K-C inequality . . . 95
7.5 Disaggregate K-$C_{02}$ inequalities for the DC-CEARP . . . . . 97
7.6 Standard Path-Bridge for the DC-CEARP . . . . . 99
7.7 Impact of the connectivity inequalities. . . . . 114
7.8 Impact of the parity inequalities. . . . . 115
7.9 Impact of the max-distance inequalities . . . . . 116
7.10 Impact of the K-C, K-$C_{02}$ and Path-Bridge inequalities. . . . . 117
7.11 Performance profile . . . . . 120

8.1 PP's solutions with subtours that satisfy connectivity inequalities . . . 138
8.2 Instances per vehicle solved optimally. . . . . 151
8.3 Performance profiles for different number of vehicles. . . . . 152

# List of Tables


5.1 Characteristics of the sets of instances . . . . . 64
5.2 Average min/max profit per instance set and profit interval . . . . . 65
5.3 Heuristic and B&C results in all instances . . . . . 65
5.4 Heuristic and B&C results with profit interval $0.65\mu$ - $1.05\mu$ . . . . . 66
5.5 Heuristic and B&C results with profit interval $0.80\mu$ - $1.20\mu$ . . . . . 66
5.6 Heuristic and B&C results with profit interval $0.95\mu$ - $1.35\mu$ . . . . . 67
5.7 B&C results, grouped by interval profits, for the largest instances . . . 68

6.1 Characteristics of the instances . . . . . 77
6.2 Results of *Matheuristic 1* for the instances with optimal solution . . . . 79
6.3 Results of *Matheuristic 2* for the instances with optimal solution . . . . 79
6.4 Results for the 30 instances with unknown optimal solution . . . . . 80
6.5 Overall results for the unsolved instances . . . . . 81
6.6 Results with and without the exact route optimization phase . . . . . 81
6.7 Impact of the optimization phase on the instances with optimal solution 81

7.1 Inequalities and separation procedures . . . . . 111
7.2 Results on the subset of 48 instances - connectivity . . . . . 114
7.3 Results on the subset of 48 instances - parity . . . . . 115
7.4 Results on the subset of 48 instances - max-distance . . . . . 116
7.5 Results on the subset of 48 instances - K-C, K-$C_{02}$ and Path-Bridge . . 117
7.6 Characteristics of the instances . . . . . 118
7.7 Computational results grouped by number of vehicles and customers . . 120
7.8 Results on the 11 instances not solved by any algorithm . . . . . 121
7.9 Results on the 27 instances not solved by at least one algorithm . . . . 121
7.10 Results for the Random50 instances . . . . . 122
7.11 Results for the Random75 instances . . . . . 122
7.12 Results for the Albaida instances . . . . . 123
7.13 Results for the Madrigueras instances . . . . . 123

8.1 Characteristics of the MM-CEARP instances . . . . . 149
8.2 Number of instances grouped by dataset and number of vehicles . . . . 149
8.3 Summary of the results on the 258 instances. . . . . 149
8.4 Computational results grouped by number of vehicles . . . . . 150
8.5 Gap comparison . . . . . 153

# List of Problems

| | |
|---|---|
| **AMRSTP** | Automatic Meter Reading Shortest Tour Problem |
| **ARP** | Arc Routing Problem |
| **CARP** | Capacitated Arc Routing Problem |
| **CEARP** | Close-Enough Arc Routing Problem |
| **CETSP** | Close-Enough Travelling Salesman Problem |
| **CPP** | Chinese Postman Problem |
| **CTP** | Covering Tour Problem |
| **DC-CEARP** | Distance-Constrained Close-Enough Arc Routing Problem |
| **DCPP** | Directed Chinese Postman Problem |
| **DGRP** | Directed General Routing Problem |
| **DRPP** | Directed Rural Postman Problem |
| **GARP** | Generalized Arc Routing Problem |
| **GDRPP** | Generalized Directed Rural Postman Problem |
| **GRP** | General Routing Problem |
| **GCSP** | Geometric Covering Salesman Problem |
| **GTSP** | Graphical Traveling Salesman Problem |
| **MM-CEARP** | Min-Max Close-Enough Arc Routing Problem |
| **MCPP** | Mixed Chinese Postman Problem |
| **MCRP** | Mixed-Constrained Routing Problem |
| **MGRP** | Mixed General Routing Problem |
| **MRPP** | Mixed Rural Postman Problem |
| **NRP** | Node Routing Problem |
| **NRPP** | Node Routing Problem with Profits |
| **PCEARP** | Profitable Close-Enough Arc Routing Problem |
| **PARP** | Prize-Collecting Arc Routing Problem |
| **PRPP** | Profitable Rural Postman Problem |
| **PRPPMV** | Profitable Rural Postman Problem with Multiple Visits |
| **RPP** | Rural Postman Problem |
| **SCEARP** | Stochastic Close-Enough Arc Routing Problem |

| | |
|---|---|
| **SCP** | Set Covering Problem |
| **TSP** | Travelling Salesman Problem |
| **TSPN** | TSP with Neighborhoods |
| **VRP** | Vehicle Routing Problem |
| **WPP** | Windy Chinese Postman Problem |
| **WGRP** | Windy General Routing Problem |

# List of Acronyms

| | |
|---|---|
| **B&C** | Branch and Cut |
| **B&P** | Branch and Price |
| **COP** | Combinatorial Optimization Problem |
| **GRASP** | Greedy Randomized Adaptive Search Procedure |
| **ILP** | Integer Linear Problem or Programming |
| **ILS** | Iterated Local Search |
| **K-C** | K-C inequalities |
| **LHS** | Left Hand Side |
| **LMP** | Linear Master Program |
| **LP** | Linear Problem or Programming |
| **MIP** | Mixed Integer Problem or Programming |
| **MILP** | Mixed Integer Linear Problem or Programming |
| **MP** | Master Program |
| **P-B** | Path-Bridge inequalities |
| **PP** | Pricing Problem |
| **RHS** | Right Hand Side |
| **RFID** | Radio Frequency Identification |
| **RLMP** | Reduced Linear Master Program |

# Chapter 1

# Introduction

## 1.1 Motivation: Transportation and Routing Problems

Transportation sector has historically represented a key driver for the economic and social development of society, being transport infrastructures the connection between citizens, employment, education, and services. For instance, rural roads enabled the access to health care and the schooling of children living in rural areas, as well as the increase and diversification of farmers' incomes by better connecting them to markets. Altogether, it has resulted in a global network that facilitates the provision of goods and services around the world, and makes people interaction easier. Far from being a finished project, the world is continuously urbanizing and provides the opportunity to create safer, cleaner and more efficient transportation systems.

Although there is no specific data, it is estimated that transport accounts for about 64% of global fuel consumption, 27% of total energy consumption and 23% of global energy-related carbon dioxide ($CO_2$) emissions. Moreover, globalization has removed barriers by making accessibility possible to all places, products and services around the world, so the environmental impact of the transport sector is expected to increase dramatically. This is why transportation is at the center of key development challenges to boost prosperity and thus achieve sustainable development that reduces energy consumption. To this end, the technology developed for the analysis and exploitation of the huge amount of data being generated by sources such as sensing data, Internet data or user behavior, are helping to define travel patterns and needs, improving the quality and efficiency of transport solutions.

At the same time, consumer behaviors and expectations are evolving because access to new technologies has caused, among many other things, a growing trend towards immediacy in society. The transport system has always been a crucial factor for companies that transport goods or provide services, and has now become a differentiating element in their efforts to maximise customer satisfaction. For this reason, transportation is not just about moving goods from one place to another or providing a specific service, but is a strategic process that seeks to reduce logistics costs and improve the customer experience. It is now a priority for companies to adopt an optimal quality of service that provides greater efficiency in timing and adaptability, safety, and resilience.

Figure 1.1: Transport infrastructure.

Transportation logistics can be optimized through proper route planning to maximize productivity, save on transportation costs, and boost profitability. Exorbitant amount of money is being spent on fuel, vehicle operation, maintenance and labor. Therefore, after years of learning about the role and importance of logistics, it is increasingly considered key to optimize it and turn it into a competitive advantage. A small improvement in routing problems can lead to huge logistics savings in absolute terms. It is estimated that the use of computerized procedures for the distribution process planning produces substantial savings (generally from 5% to 20%) in the global transportation costs. Indeed, the transportation process involves all stages of the production and distribution systems and represents generally from 10% to 20% of the goods final cost.

In the academic world, route optimization is referred as the determination of the most cost-efficient route taking into account relevant factors involved such as vehicle limitations, cost controls, time windows, resource limitations concerning the loading

process at the depot, etc. Routing problems are then defined as the design of optimal routes from a depot to a set of destinations with specific constraints. These problems have attracted the attention of many researchers and practitioners during the last 60 years because of the mathematical challenges involved in their study and solution, and the motivation in the big economic impact that has the improvements found. Many types of routing problems are differentiated according to the type of goods or service to be performed, the characteristics of the vehicle fleet (size, capacity, autonomy), the distance and level of service of the customers, whether route readjustment is allowed on the fly, etc.

Most routing problems are NP-*hard* and, as they are extremely difficult to solve optimally in practice, they are classified according to the specifications of the real-life situations to be modeled. In Chapter 3, it will be shown the classification of those routing problems in which customers can be represented by nodes on a graph (node routing problems, NRPs), and those ones in which the service is performed on the arcs or edges (arc routing problems, ARPs). Although the research on routing problems has traditionally been focused more on NRPs, the literature on ARPs is growing every day, and the number and efficiency of algorithms designed for these problems have increased considerably in recent years. This thesis is focused in the framework of the ARPs.

The origin of the arc routing problems lies in the famous problem of the Seven Bridges of Königsberg. The city of Königsberg (Russia), crossed by the river Pregel, was made up of two large islands connected to each other and to the two banks of the river by seven bridges (Figure 1.2). It seems that the citizens of Königsberg wondered if it was possible to find a tour that visited all parts of the city and crossed each bridge exactly once. It was in 1736 that Euler proved that no such route existed.

However, it was not until many years later, in the 1960s, when the first proper arc routing problem was introduced. It was a problem that modeled a situation in which the service was carried out along the streets of a city and tried to optimize the length of the route. This ARP proposed by Guan (1962), the Chinese Postman Problem (CPP), was defined as the design of a route traversing all the arcs and/or edges requiring service of a given graph. It seems that the name of the problem is due to the fact that the author was Chinese and it addressed the situation encountered by a postman to find a path with minimum length for the mail delivery. Given a graph, the CPP aims to find a closed minimum-cost path that traverses all the arcs and/or edges at least once. Subsequently, it was proposed the Rural Postman Problem (RPP), a generalization of the CPP in which services did not have to be performed on all streets. The RPP objective is also to find a closed path with minimum cost that

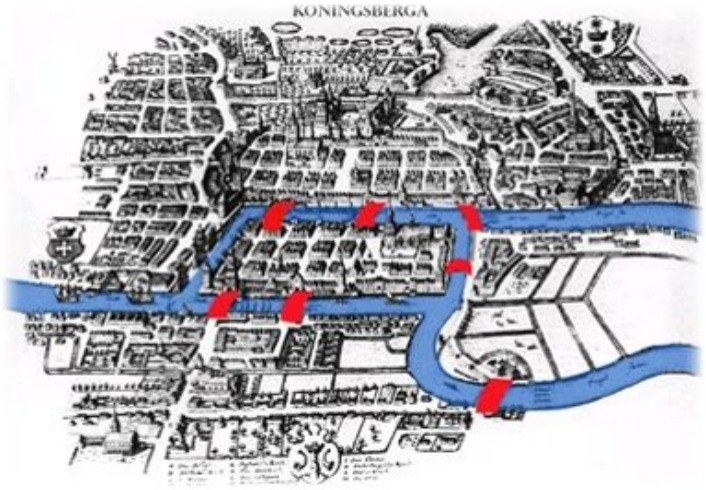



Figure 1.2: The Seven Bridges of Königsberg.

traverses at least once not all the arcs and/or edges, but only those requiring to be serviced. From this point on, a large number of variants of these problems have been studied in detail in the literature.

In addition to mail delivery, arc routing problems have many applications in organizing tasks such as garbage collection, snow ploughing, milk delivery, inspection of distribution systems (electricity, telephone or railroad networks), street cleaning and watering, etc. Millions of euros are spent each year on these operations and the savings achieved by optimizing them is enormous. The main challenge with these problems is that they cannot be modeled as simple arc routing problems, but each problem has its own characteristics. Therefore, the methodology used to solve them must be specific and must consider the context of each problem.

## 1.2 Scope of this thesis: CEARP

Relevant technological innovations, such as new types of devices, radio frequency identification technology (RFID), and the availability of real-time data through geolocation, traffic flows or communication between customers and drivers, have led to numerous changes in the business of transportation logistics. At the same time, Operations Research has continued to evolve and this has required the definition and study of new problems, as well as the incorporation of new features to existing ones.

In particular, the development of new technologies has given rise to scenarios in which the vehicle is not required to arrive at the customer's location in order to perform the service, but rather that it is sufficient to approach it. In routing problems this feature is called "close enough", since the vehicle only needs to pass close enough

to the customer's position to perform the service. If, in the problem under study, customer service is modeled or performed on the arcs of a graph, we obtain what is known as the Close Enough Arc Routing Problem (CEARP). This problem consists of finding a minimum cost route that starts and ends at a depot and traverses some of the streets of a road network in such a way that all customers are serviced.

Figure 1.3: Automatic Meter Reading.

One of the most direct applications of CEARP is in the automatic meter reading, since nowadays the operators of gas, water and electricity companies do not need to traverse all the streets of their customers to read the consumption of their meters. Other relevant applications of CEARP are, for example, inventory management in department stores, where drones are being used that only need to pass close enough to the products to inventory them, in network surveillance and maintenance tasks, as well as in robotic monitoring of wireless sensor networks. All these applications are explained in detail in Chapter 4.

This thesis focuses on three extensions of the CEARP: the Profitable CEARP, for a single vehicle, and the Distance-Constrained CEARP and the Min-Max CEARP, for multiple vehicles. In the first problem, the Profitable CEARP, each customer has associated a profit that is collected if the customer is serviced. In this problem not all customers have to be serviced, but only those more interesting from the point of view of the profit provided. In the second problem, the Distance-Constrained CEARP, the service is done by a fleet of vehicles with a the maximum length or time for their routes, and the objective is to minimize the total traversed length. Finally, the Min-Max CEARP also considers a fleet of vehicles to service all the customers but in this

case the length of the longest route is minimized in order to balance the length of the planned routes.

## 1.3 Outline

This thesis is divided into nine chapters as follows:

**Chapter 1. Introduction.** This chapter gives the reader a brief overview of transport logistics, contextualizing and motivating the development of the thesis. We present the importance of route optimization, in particular arc routing problems, in logistic procedures. We refer to the origin of arc routing problems, showing how they have evolved along with new technologies, list the specific problems that have been studied, and finally provide an outline of the thesis.

**Chapter 2. Preliminaries.** In this chapter, some concepts of Mathematical Programming are described in order to provide the readers with a summarized conceptual review and to introduce the topics, the terminology, and the mathematical notation used throughout the thesis. Several basic principles of graph theory, linear and integer programming, and polyhedral theory and polyhedral combinatorics are here presented.

**Chapter 3. Routing Problems.** This chapter provides an overview of some classic routing problems. In addition, it presents the basic characteristics of many other related problems that have arisen over the years to study specific situations.

**Chapter 4. Close-Enough Routing Problems.** We present in this chapter the class of routing problems on which this thesis is focused and some real-world applications. We first summarize the state of the art of the close-enough problem in a node routing context, the Close Enough Traveling Salesman Problem. Then, in more detail, we provide a literature review of the problem for the arc routing context, the CEARP, as well as the formal definition of the problem, a formulation and a set of valid inequalities. Finally, we comment some variants of CEARP.

**Chapter 5. The Profitable CEARP.** This chapter starts formally defining the Profitable CEARP, proposing two different formulations and some valid inequalities, and studying its polyhedron of solutions. To solve this problem, we present a heuristic algorithm and a branch-and-cut algorithm that includes the separation procedures for the identification of violated inequalities. Extensive computational experiments

on four randomly generated sets of instances are provided to show the performance of the proposed algorithms. The chapter is the outcome of a collaboration with Prof. Bianchessi, from the Università degli Studi di Milano, the destination of my research visit. The following paper has been submitted to an international journal for publication:

> N. Bianchessi, Á. Corberán, I. Plana, M. Reula, J.M. Sanchis. 2021. The Profitable Close Enough Arc Routing Problem. Under revision.

**Chapter 6. A Matheuristic for the Distance-Constrained CEARP.** In this chapter the Distance-Constrained CEARP is studied, for which a multi-start matheuristic is proposed that incorporates an effective branch-and-cut method for the CEARP in order to optimize the routes obtained. First, the DC-CEARP is formally defined and the most promising formulation is given. Then, a matheuristic to solve this problem is presented together with the computational results obtained. The chapter is based on the following published paper:

> Á. Corberán, I. Plana, M. Reula, J.M. Sanchis. 2019. A matheuristic for the Distance-Constrained Close Enough Arc Routing Problem. **TOP**. 27, 312–326.

**Chapter 7. On the Distance-Constrained CEARP.** This chapter contains a deepen study of the DC-CEARP. A new formulation for the DC-CEARP is proposed that combines the best features of the previously existing ones, its associated polyhedron is studied, and several families of valid inequalities are presented. Moreover, we present an efficient branch-and-cut algorithm based on the separation of the new inequalities. An extended computational analysis has been carried out in which the efficiency of the proposed algorithm is proved. The chapter is based on the following published paper:

> Á. Corberán, I. Plana, M. Reula, J.M. Sanchis. 2021. On the Distance-Constrained Close Enough Arc Routing Problem. **European Journal of Operational Research.** 291(1), 32-51.

**Chapter 8. The Min-Max CEARP.** In this chapter we introduce the MM-CEARP, focusing on its modeling and exact solution. Two different models for the problem are presented: an arc-based formulation making use of arc and servicing variables, and a route-based set covering formulation. Solution algorithms to solve the problem are then presented: a B&C algorithm addressing the arc-based formulation

and a B&P algorithm based on the set covering one. A heuristic used to compute solutions with which initializing the exact algorithms is described. The exact algorithms are compared in benchmark instances through extensive computational experiments. The chapter is also the outcome of a collaboration with Prof. Bianchessi, from the Università degli Studi di Milano, the destination of my research visit. The following paper has been submitted to an international journal for publication:

> N. Bianchessi, Á. Corberán, I. Plana, M. Reula, J.M. Sanchis. 2021. The Min-Max Distance-Constrained Close Enough Arc Routing Problem. Under revision.

**Chapter 9. Conclusions and future work.** This chapter presents the contributions of the thesis and outlines the lines of research and future work that could be derived from it.

# Chapter 2

# Preliminaries

In this chapter, we describe some essential concepts of Mathematical Programming to provide non-expert readers with a detailed conceptual review and to introduce the topics, the terminology, and the mathematical notation used in the following chapters. We will present several basic principles of graph theory, linear and integer programming, and polyhedral theory and polyhedral combinatorics.

## 2.1 Graph Theory

In this section, we introduce some definitions and results of graph theory that we have used it throughout the thesis. Graphs are often used to model many real-life problems, such as arc routing problems. Using a graph, these problems can be represented clearly and precisely, although a priori the problem may seem complex and imprecise. Some classical references in this area are Harary (1969), Berge (1973), Bondy and Murty (1976) and Christofides (1975).

A **graph** $G$ is defined as a pair $(V, E)$ in which the elements of $V$ are known as vertices or nodes and the elements of $E$ are pairs of vertices that are called edges or arcs. Depending on whether the pairs of vertices are sorted or not, the graphs can be classified as follows:

- *Undirected:* $E$ is a set of non sorted pairs, i.e., it only contains edges. It is denoted as $G = (V, E)$.
- *Directed:* $E$ is a set of sorted pairs, i.e., it is formed only by arcs. In this case $G$ is denoted as $G = (V, A)$.

- *Mixed:* $E$ consists of sorted and non sorted pairs. In this case, the set of non sorted pairs (edges) is denoted by $E$ and $A$ represents the set of ordered pairs (arcs). We therefore denote $G = (V, E, A)$.

In general, we will represent the edges and the arcs as pairs of vertices $(i, j)$, and will denote by $n = |V|$ the number of vertices, by $m_E = |E|$ the number of edges, and by $m_A = |A|$ the number of arcs. If the graph only contains arcs or edges, its number will be simply denoted by $m$. The edges and arcs that start and end at the same vertex $i \in V$, $(i, i)$, are called *loops*. Two edges are *parallel* if they join the same vertices. In a directed graph, two arcs are said to be *parallel* if they have the same initial and final vertex.

Graphs without loops nor edges or arcs in parallel are called **simple graphs**. An undirected simple graph in which there is an edge between each pair of vertices is called a **complete graph**. A directed simple graph is **complete** if there are two arcs $(i, j)$ and $(j, i)$ between any pair of vertices $i, j \in V$. Complete graphs are commonly designated $K_n$, where $n$ is the number of vertices. A graph $G$ is **bipartite** if there is a partition of $V = X \cup Y$, with $X, Y$ non empty, so that each edge or arc of the graph has a vertex in $X$ and another in $Y$.

The **degree** of a vertex $v \in V$ is the number of edges and arcs incident with it, and is represented by $d(v)$. We say that a vertex is even when its degree is even and odd otherwise. In a directed graph, the number of arcs with $v$ as the final vertex is its **indegree** and is represented by $d^-(v)$. Similarly, the **outdegree** of a vertex $v$, $d^+(v)$, is the number of arcs with $v$ as the initial vertex.

**Theorem 1.** *Let $G$ be an undirected graph. The number of vertices with odd degree is always an even number.*

If $S_1$ and $S_2$ are two disjoint subsets of $V$, we denote by $(S_1 : S_2)$ to the set of edges and arcs with a vertex in $S_1$ and the other in $S_2$. In particular, the set $(S : \overline{S})$, where $\overline{S} = V \setminus S$, is usually represented by $\delta(S)$ and is called the **cutset** associated with $S$. A cutset $\delta(S)$ is even (resp. odd) if the number of edges and arcs it contains is even (resp. odd). In particular, $\delta(\{i\})$ (abbreviately $\delta(i)$), $i \in V$ , is a cutset of $G$.

**Theorem 2.** *$\delta(S) = (S : \overline{S})$ is an odd cutset of $G$ if and only if $S$ and $\overline{S}$ contains an odd number of odd vertices.*

If $G = (V, E, A)$ and $G' = (V', E', A')$ are two graphs such that $V' \subseteq V, E' \subseteq E$, and $A' \subseteq A$, we say that $G'$ is a **subgraph** of $G$. Given a subset $V' \subseteq V$, $E(V')$ and $A(V')$ denote the set of edges and arcs of $G$ with both ends in $V'$, and $G(V') = (V', E(V'), A(V'))$ is the **subgraph of $G$ induced by $V'$**. Similarly,

$G(A') = (V(A'), A')$ is the subgraph of $G$ induced by $A' \subseteq A$ and $G(E') = (V(E'), E')$ the subgraph induced by the set of edges $E' \subseteq E$.

A **path** of length $k$ is a sequence $w = (i_0, e_1, i_1, e_2, \ldots, e_k, i_k)$ in which the vertices and edges (or arcs) appear alternately in such a way that the initial and the final vertex of any edge (or arc) $e_r$ are $i_{r-1}$ and $i_r$, respectively. If we deal with simple graphs, a path can be represented only by the sequence of vertices $w = (i_0, i_1, \ldots, i_k)$ or by the sequence of edges $w = (e_1, e_2, \ldots, e_k)$. A path where all the vertices are different is called a **simple path**. If a path starts and ends at the same vertex is called a **tour** (or **closed path**).

In many problems defined on graphs, it is convenient to define a function that assigns a non-negative value to each edge or arc depending on its length or in the cost of traversing it. Therefore, given a graph $G = (V, E, A)$, we define a *cost function*:

$$\begin{aligned} \mathcal{C} : E \cup A &\longrightarrow \mathbb{R} \\ (i, j) &\longrightarrow c_{ij} \geq 0. \end{aligned}$$

The total cost or length of a path is the sum of the costs associated with the edges and arcs that defining it.

**Theorem 3.** *If $w$ is a tour in $G$, then $w$ traverses an even number (or zero) the edges and/or arcs of any cutset in $G$.*

Given an undirected graph $G$, we say that two vertices are connected if and only if there is a path between them. In the case of a directed graph, it is required that, in addition to the existence of a path from $i$ to $j$, there is another path from $j$ to $i$. A graph $G$ is **connected** if, given any pair of vertices, there is at least one path between them. A mixed or directed graph is **strongly connected** if, for any pair of vertices $i$ and $j$, there is a path from $i$ to $j$ and another from $j$ to $i$. A **connected component** of $G$ is a maximal connected subgraph of $G$. The connected components of $G$ define a partition of $V$.

Focusing on the traversals of the edges and arcs of a graph, a **Eulerian tour** in $G$ is a tour that traverses each edge and arc exactly once. A graph containing a Eulerian tour is called a **Eulerian graph**. A **Hamiltonian tour** is a tour that visits all the vertices of the graph exactly once and a graph that contains a Hamiltonian tour is called a **Hamiltonian graph**.

**Theorem 4.** *A connected and undirected graph $G$ is Eulerian if, and only if, the degree of all the vertices is even.*

**Theorem 5.** *A strongly connected directed graph $G$ is Eulerian if, and only if, the indegree and outdegree of each vertex are equal.*

## 2.2 Linear and Integer Programming

**Linear Programming** was first conceived by George B. Dantzig around 1947, while he was working as a mathematical advisor to the Comptroller of the United States Air Force on developing a mechanized planning tool for a time-staged deployment, training, and logistical supply program. Linear problems (LPs) deal with a set of *decision variables* $x \in \mathbb{R}^n$ satisfying a finite set of *constraints*, which are used to minimize/maximize a linear cost function $f(x) = c^T x,\ c \in \mathbb{R}^n$, known as **objective function**. The set of constraints, given as linear inequalities $ax \leq \alpha$ with $a \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$, generates the *feasible region* $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ containing all **feasible solutions**. An **optimal solution** $x^* \in P$ satisfies:

$$c^T x^* = \min\{c^T x : x \in P\}.$$

In the early 50's, Dantzig proposed an algorithm called the **simplex method** to solve LPs. It has been widely accepted because of its ability to model complex management decision problems and its capability for producing solutions in a reasonable amount of time. This algorithm is non-polynomial because it may require a number of operations that depends exponentially on the size of the instance. Since the simplex method was introduced, there have been many contributions in the area of linear programming such as theoretical developments, computational aspects and exploration of new applications of the subject. Among them, the most important contribution in Linear Programming was made by Khachian (1979). In that paper, a polynomial algorithm (the **ellipsoid method**) to solve LPs in polynomial time was proposed. Therefore, it was proved that LPs are problems in the class $P$.

Dealing with large-scale LPs, Dantzig and Wolfe (1960) developed a decomposition principle which is a procedure for solving linear programs with a large number of variables or linear programs that contain specially structured constraints. The original linear problem is reformulated and the new constraints are divided into *general* or *linking constraints*, and *special* or *independent constraints*. The strategy of the decomposition procedure is to operate on two separate linear problems: the *master problem*, over the set of general constraints; and the *subproblem* or *pricing problem*, over the set of special constraints. First, the master problem is solved using an iterative column generation algorithm that solves at each iteration a restricted master problem, that is, the master problem restricted to a subset of the variables that varies from one

iteration to another. Then, the subproblem, which does not necessarily have to be the same every time, is solved with a different objective function at each iteration. The iterative procedure is finite and finishes when the optimal solution of the master problem is found.

In many applications of optimization, one would really like the decision variables to be restricted to integer values. When it occurs, the feasible region becomes a discrete set of points in $\mathbb{R}^n$. This fact may suggest that integer programs are a simple subject, but they can turn certifiably hard. **Integer Linear Problems** (ILPs) consist of minimizing (or maximizing) a linear function with integer decision variables $x \in \mathbb{Z}^n$ on the feasible region $P = \{x \in \mathbb{Z}^n : Ax \leq b\}$. The ILPs are frequently identified with Combinatorial Optimization, which concerns the study of optimization problems of combinatorial nature. In fact, most ILPs are considered NP-*hard* problems, since the difficulty of optimizing on integer variables requires a methodology according to their complexity. If the integrity constraint of an ILP is omitted, it results in an LP known as **linear relaxation** of the ILP.

Whenever we have an optimization problem, it is solved when we have reached the optimal solution. However, in practice it is not always necessary to reach the optimal solution of the problem. Since the problems are sometimes very difficult to solve, a non-optimal solution in a shorter time may be preferred. We distinguish between two types of solution algorithms. Those in which we look for the optimal solution, the **exact algorithms**, and those in which we cannot guarantee to find the optimal solution, but we can achieve a high-quality one in a reasonable time, the **approximation** or **heuristic algorithms**.

Among the exact algorithms, *Branch and bound* is a smart and systematic search for the optimal integer solution. In a branch-and-bound scheme, the linear relaxation of the original problem is solved at the root node. Then, by branching on the integer values of the variables, problems are hierarchically generated on a tree. Optimal values of the linear relaxations at the nodes are used as bounds to prune the tree, until the optimal integer solution is reached. A generalization of this procedure is the *Branch and cut*. In the nodes of the tree, a standard but not trivial procedure improves the bounds by adding general or problem specific inequalities. At each node, the new inequalities are introduced into a pool of constraints that will be managed by the solver. The number of inequalities to add can be exponential and not carefully designed cutting schemes can easily get out of hand. As we describe in Section 2.3, the separation algorithms used to solve the separation problems associated with each family of valid inequalities allow us to find inequalities not satisfied by the LPs solutions that may improve the search for the optimal solution.

Dealing with integer programming problems that allow the application of the principle of decomposition for the LPs, it is possible to solve them by means of a *branch-and-price* algorithm. It is more sophisticated and generally shows a good performance in large-scale problems. The algorithm begins by using a reformulation to define the master problem. Then, it solves the linear relaxation of a restricted master problem in which only a subset of the columns is considered. To check for optimality, a pricing problem is solved to find columns that can improve the objective function. Each time a column is found with negative reduced cost, it is added to the restricted master problem and the relaxation is reoptimized. If no columns can enter the basis and the solution to the relaxation is not integer, then branching occurs until the solution is integer.

Alternatively to the exact methods we have the heuristics algorithms. A large number and variety of difficult problems which appear in practice and need to be solved consciously, encourage the development of efficient procedures to find good solutions even if they are not optimal. In these methods, the speed of the process is considered as important as the quality of the solution obtained. Heuristics containing high-level procedures are referred in the literature as *metaheuristic*. Finally, the maturation of the subject has evolved into *matheuristics* which, as its name suggests, are the hybridization of exact methods with heuristics. Both types of algorithms interoperate until some stop criterion is reached. In spite of using exact methods, the solution reached does not have to be optimal.

The following references are some excellent texts of Linear and Integer Programming: Dantzig (1963), Garfinkel and Nemhauser (1972), Bazaraa and Jarvis (1977), Chvátal (1983), Schrijver (1986) and Nemhauser and Wolsey (1988).

## 2.3 Polyhedral theory and polyhedral combinatorics

In this section we summarize the basic ideas of the polyhedral approach to solve combinatorial optimization problems (COPs), especially those that are NP-*hard*. First, we review some theoretical concepts and establish the terminology we use.

Let us denote by $\mathbb{R}^n$ all the column vectors with $n$ real components and let $x_i \in \mathbb{R}^n, i = 1, \ldots, m$. A vector $y \in \mathbb{R}^n$ is a **linear combination** of $x_i$ if, and only if, $y$ can be written as $y = \sum_{i=1}^{m} \lambda_i x_i$, with $\lambda_i \in \mathbb{R}$. In addition, if $y$ is a linear combination in which $\lambda_i \geq 0$, then $y$ is a **conic combination** of the vectors $x_i$. A vector $y$ is an **affine combination** if it is a linear combination that satisfies $\sum_{i=1}^{m} \lambda_i = 1$. Finally, if $y$ is an affine and conic combination, then it is a **convex combination**.

The **linear** (resp. **conic, affine, convex) hull** of a set $X \subseteq \mathbb{R}^n$ is the set of all points that are linear (resp. conic, affine, convex) combinations of a finite number of points of $X$. It is denoted by $lin(X)$ ($con(X)$, $aff(X)$, $conv(X)$, respectively). $X$ is a **linear** (resp. **conic, affine, convex) subspace** in $\mathbb{R}^n$ if $X$ is equal to its linear (resp. conic, affine, convex) hull.

Vectors $x_i \in \mathbb{R}^n$, $1 \leq i \leq m$, are **linearly independent** if none of them is a linear combination of the others, or equivalently if $\sum_{i=1}^m \lambda_i x_i = 0$ implies that $\lambda_i = 0$ for all $i$. Otherwise, we say that they are **linearly dependent**. In the same way, we say that the vectors $x_i \in \mathbb{R}^n$, $1 \leq i \leq m$ are **affinely independent** if none of them is an affine combination of the others, or equivalently if $\sum_{i=1}^m \lambda_i x_i = 0$ with $\sum_{i=1}^m \lambda_i = 0$, implies $\lambda_i = 0$ for all $i$. Otherwise, we say that the vectors are **affinely dependents**.

**Theorem 6.** *For a set $X \subseteq \mathbb{R}^n$ the following are equivalent:*

- *$X$ is affinely independent.*
- *Given $y \in X$, $\{x - y : \ x \in X, x \neq y\}$ is linearly independent.*
- *Given $y \in \mathbb{R}^n$, $\{x - y : \ x \in X\}$ is affinely independent.*

Given $X \subseteq \mathbb{R}^n$ the **range** of $X$ is the maximum number of linearly independent vectors in $X$ and is denoted by $rg(X)$. In the same way, the **affine range** of $X$ is the maximum number of affinely independent vectors of $X$ and is denoted by $rg_a(X)$. The range of a matrix $A$, $rg(A)$, is the range of its column vectors, which is the same as the range of its row vectors.

**Theorem 7.** *Given $X \subseteq \mathbb{R}^n$*

- *If $0 \in aff(X)$, then $rg_a(X) = rg(X) + 1$.*
- *If $0 \notin aff(X)$, then $rg_a(X) = rg(X)$.*

If $X$ is a linear subspace of $\mathbb{R}^n$, a **base** $B$ of $X$ is any finite subset of linearly independent vectors in $X$ such that $lin(B) = X$. The bases of the same linear subspace have the same number of vectors, and its number is the **dimension** of $X$. If $X$ is an affinely subspace in $\mathbb{R}^n$, there is a unique linear subspace $X' \subseteq \mathbb{R}^n$ such that $X' = \{x - x^* : x \in X\}$, for any $x^* \in X$. The dimension of $X$ is the dimension of $X'$. Finally, if $X$ is an arbitrary subset in $\mathbb{R}^n$, the dimension of $X$ is the dimension of $aff(X)$ and is denoted by $dim(X)$.

A **hyperplane** of $\mathbb{R}^n$ is a set $\{x \in \mathbb{R}^n : a^T x = \alpha\}$ and a **half-space** of $\mathbb{R}^n$ is a set $\{x \in \mathbb{R}^n : a^T x \leq \alpha\}$, where $a \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$. A **polyhedron** of $\mathbb{R}^n$ is the intersection of a finite number of half-spaces in $\mathbb{R}^n$ or, equivalently, the solution set

of an inequalities system $Ax \leq b$, where $A$ is an $m$ x $n$ array and $b \in \mathbb{R}^m$. A bounded polyhedron is called a **polytope**. A polyhedron $P \subseteq \mathbb{R}^n$ is **full-dimensional** if $dim(P) = n$.

We say that $a^T x \leq \alpha$ is a **valid inequality** for a polyhedron $P$ if $P \subseteq \{x \in \mathbb{R}^n : a^T x \leq \alpha\}$. If $F \subseteq \mathbb{R}^n$, besides being a valid inequality $a^T x \leq \alpha$, satisfies that $F = P \cap \{x \in \mathbb{R}^n : a^T x = \alpha\}$, $F$ is a **face** of the polyhedron $P$. Here we say that the face $F$ is induced by the inequality $a^T x \leq \alpha$. Several inequalities can induce the same face of a polyhedron $P$. In this case, these inequalities are equivalent with respect to $P$. Obviously, a polyhedron $P$ is a face of itself. **Proper faces** are the other non-empty faces of a polyhedron. Likewise, a **facet** of a polyhedron is any non-empty proper face of $P$ that is maximal with respect to the inclusion of sets. In this way, the concept of facet corresponds to the common idea of a polyhedron's face. Facets are then the best valid inequalities that can be added to a formulation. In an ideal theoretical scenario, all the constraints in the formulation would be facets.

**Theorem 8.** *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and let $F$ be a non-empty proper face of $P$ induced by a valid inequality $a^T x \leq \alpha$. So, the following are equivalent:*

- *$F$ is a facet of $P$.*
- *$dim(F) = dim(P) - 1$.*
- *If $b^T x \leq \beta$ is valid for $P$, $F \subseteq P \cap \{x \in \mathbb{R}^n : b^T x \leq \beta\}$ and $aff(P) = \{x \in \mathbb{R}^n : Ax = d\}$, where $A$ is a matrix $mxn$ and $d \in \mathbb{R}^m$, then there are $\lambda \in \mathbb{R}^n$ and $\mu \geq 0$ such that $b^T = \mu a^T + \lambda^T A$.*

A **vertex** of a polyhedron $P$ is any non-empty proper face of $P$ that is minimal with respect to the inclusion of sets, that is, any face of $P$ that is constituted by a single point $F = \{v\}$. We call **integer polyhedron** to any polyhedron whose vertices have all its components integer.

**Theorem 9.** *$v \in P$ is a vertex if, and only if, $v$ cannot been expressed as a convex combination of other points in $P$.*

**Theorem 10.** *If a polyhedron $P$ has at least one vertex and $\min\{c^T x : x \in P\}$ is finite, then there is at least one vertex in $P$ which is an optimal solution.*

**Theorem 11.** *For each vertex $x^* \in P$, there is a vector $c \in \mathbb{R}^n$ such that $c^T x^* < c^T x$ for each $x^* \in P \setminus \{x\}$.*

Given a finite set $E$, called *ground set*, with a cost function $c$, and a finite (or infinite numerable) family $\mathcal{F}$ of subsets of $E$, called feasible solutions, a **combinatorial optimization problem** (COP) with linear objective function consists of finding an $F^* \in \mathcal{F}$ such that $c(F^*) = \sum_{e \in F^*} c_e x_e$ is minimum (or maximum), where $x_e$ denotes the number of times that $e \in E$ is in $F^*$.

The polyhedral approach to the solution of combinatorial optimization problems begins with the creation of a polyhedron $P_{\mathcal{F}}$, whose vertices, possibly together with other points in $P_{\mathcal{F}}$, follow a one-to-one correspondence with the feasible solutions in $\mathcal{F}$. We define the incidence vector $x^F$ of a feasible solution $F$ in $\mathcal{F}$ as $x^F = (x_e^F)_{e \in E} \in \mathbb{Z}^{|E|}$, where $x_e^F$ denotes the number of times that $e$ is in $F$, and then we define $P_{\mathcal{F}}$ as $P_{\mathcal{F}} = conv\{x^F : F \in \mathcal{F}\}$.

Although $P_{\mathcal{F}}$ is not a polyhedron in general, it can be proved that for most COPs it actually is. Thus, we will assume that $P_{\mathcal{F}}$ is a polyhedron. It can be seen that each possible solution $F$ in $\mathcal{F}$ corresponds to a point in $P_{\mathcal{F}}$ and each vertex in $P_{\mathcal{F}}$ is a solution in $\mathcal{F}$. Therefore, for any cost function $c$ for which the COP has a finite optimum value, we can find an optimal solution for the COP by optimally solving the LP:

$$\min_{x \in P_{\mathcal{F}}} cx \tag{2.1}$$

What we have done is to transform in a natural way the COP into the LP (2.1). However, without any knowledge of the linear system that $P_{\mathcal{F}}$ describes, this transformation is useless from the algorithmic point of view. In order to tackle the problem (2.1) by using linear programming algorithms, it is necessary to know all (or a significant part of) the linear system described by $P_{\mathcal{F}}$.

An ideal theoretical situation would be to know a complete linear description of the polyhedron, even though we would not be able to generate and keep all the data of the associated LP. Even if the linear system is finite, it can be expected to be extremely large because, in many problems, the number of rows grows exponentially with the number of variables in the problem. Bringing polyhedral theory from principles to practice requires additional specific strategies and methods. An alternative to listing all the known rows of the linear system is to periodically solve a separation problem.

The **Facet Identification Problem**, also known as **Separation Problem**, consists of, given $\overline{x} \in \mathbb{R}^{|E|}$ and a polyhedron $P_{\mathcal{F}}$, finding a linear inequality $f\overline{x} \leq f_0$ defining a facet of $P_{\mathcal{F}}$ that is violated by $\overline{x}$ (i.e., $f\overline{x} > f_0$), or demonstrate that such inequality does not exist ($\overline{x} \in P_{\mathcal{F}}$).

Note that a subroutine to exactly solve the facet identification problem embedded in a branch-and-cut code will generate violated inequalities (maybe defining facets of $P_{\mathcal{F}}$) cutting-off fractional optimal solutions of the LPs. In Grötschel and Padberg (1985) is described the *Relaxation Method*, an iterative procedure for solving combinatorial optimization problems which uses this technique to solve them. The problem is that this method requires the complete description of the linear system associated with the polyhedron $P_{\mathcal{F}}$.

The problem with the above technique is that, as we have mentioned, these complete descriptions are known only for a very few number of COPs. However, research in Polyhedral Combinatorics has demonstrated that even a partial knowledge of the linear system describing $P_{\mathcal{F}}$ generally improves results both theoretically and computationally. Thus, one of the fundamental challenges in Polyhedral Combinatorics is to find a large enough linear system describing the polyhedron $P_{\mathcal{F}}$ to provide enough information for solving the problem.

Therefore, the description of the polyhedron $P_{\mathcal{F}}$ can be used to improve the linear relaxation by adding a subset of valid inequalities. By using the **separation algorithms**, we generate inequalities of the LP from the partial description of the polyhedron. A separation algorithm is successful if it returns a valid inequality that cuts off any infeasible solution. An exact separation algorithm is guaranteed to solve the separation problem while a heuristic does not. Despite this, heuristics are frequently used in practice because they are generally faster or because the separation problem is itself a NP-*hard* problem. If applying a heuristic for solving the separation problem we do not find any violated inequality and we have not reached the optimum, we can then use a branch-and-bound or branch-and-cut scheme with a higher probability of reaching the optimal solution, since we have a much tighter linear relaxation.

Bachem and Grötschel (1982), Pulleyblank (1983), Hoffman and Padberg (1985), Schrijver (1986) and Nemhauser and Wolsey (1988) provide a wealth of information on the concepts and results of polyhedral theory and polyhedral combinatorics.

# Chapter 3

# Routing Problems

In this chapter, we provide an overview of some classic routing problems that have been studied in depth in the literature. Due to the wide variety of routing problems, we want to point out that those presented here are the basis of many other related problems that have arisen over the years to adapt them to specific situations. The knowledge of these original problems contributes to a better and faster understanding of the more complex ones. Routing problems are combinatorial optimization problems, which consist of finding the best (optimal) solution among a finite (but huge) or infinite number of numerable solutions. Usually, a problem of size $n$ may have $n$ or $k^n$ feasible solutions, where $k$ is an integer greater than or equal to 2. Therefore, to solve this type of problem it is necessary to study its properties and characteristics, which are essential to develop a tailored solution method.

Routing problems have attracted the attention of many researchers and practitioners during many years because of their big economic impact and the mathematical challenges involved in their study and solution. These problems are related to a wide range of variables, including vehicles, drivers, warehouses, roads, and customers, resulting in a large number of problem variants. In the real world, there is a large number of situations that can be modeled as routing problems. In fact, most public organizations or private companies have been dealing with problems related to mail delivery, garbage collection, street cleaning, inspection or maintenance of streets, highways or electrical networks, distribution of all kinds of products, visits to customers, transportation of people, etc. In the last years, important research has been carried out in this field, achieving significant progress in the formulation of problems as well as in the design, analysis, and implementation of algorithms for their solution. As a result of this research, it has been possible to achieve an efficient planning of the routes that helps to minimize the costs when performing these tasks.


Figure 3.1: Applications of routing problems

Usually, routing problems are modeled on a graph that represents a road network, where a non-negative cost is associated with each arc/edge of the graph (the streets or roads of the network) representing the cost/distance of traversing it. These problems can be classified according to whether the service must be performed at the vertices or the arcs of the graph. The former are called Node Routing Problems, while the latter are called Arc Routing Problems. Combining these two classes of problems, more general problems arise that consider that the service must be performed by traversing a set of edges and/or arcs and visiting a set of vertices in the graph. The "combined" problems are called General Routing Problems. Typically, in all cases, the objective is to find one or more tours with total minimum cost, visiting the vertices and traversing the arcs and/or edges of the graph that require service.

Very often, the demand from customers (nodes and/or arcs) is so great that it cannot be serviced with a single vehicle. Thus, routing problems can also be classified according to whether a single vehicle is sufficient to do the service or whether a fleet of vehicles must be considered. If there is more than one vehicle, we must take into account whether all the vehicles are identical or not. The objective of the problem is to define a route for each vehicle in such a way that the global demand is satisfied. Also, many times the routes must be balanced according to certain criteria. There are several possibilities to achieve this: to limit the capacity/distance/time of each vehicle and minimize the total distance; use a min-max objective in which the length of the longest path is minimized. If the problem requires it, a combination of both criteria can also be used.

In some situations, there is a (measurable) profit associated with customer service. Routing problems with profits refer to situations in which a profit is associated with each customer and, unlike what happens in "classic" routing problems, it is not necessary to service all customers but only a subset of them. The customers to service must be selected from a set of "potential" customers with an associated profit that is collected when the customer is serviced. They basically consist of designing one or more routes servicing the chosen customers and such that an objective defined as a function of the cost or/and the profit is optimized. Depending on the objective function, several variants can be considered. A routing problem is called *profitable* if the objective is to maximize the difference between the total collected profit and the traveling cost; *orienteering*, when the aim is to maximize the profit but the length (duration) of the route is restricted by a given distance (time); and *prize collecting* if the goal is to minimize the cost/length of the route taking into account a limitation on the profit.

Note that we have mentioned only some of the particularities that can be considered in the study of a specific problem. Any routing problem may be complemented with some additional real-life complexities, such as time windows for pickup and delivery, time-dependent travel times, precedence relations, forbidden turns, etc.

Finally, arc routing problems can be classified into undirected, directed, mixed, and "windy", depending on the characteristics of the network studied. To represent a network in which all streets or roads can be traversed in both directions at the same cost, we will use an *undirected* graph. However, the graph will be *directed* to represent a network in which streets can only be traversed in one direction, or it will be *mixed* in the case where some streets can only be traversed in one direction and others in both directions. The graph will be "windy" if we want to model a routing problem in which the streets are two-way but the cost of traversing them depends on the direction in which they are traversed (Minieka (1979)). Arc routing problems defined in a windy graph generalize those defined in undirected, directed, or mixed graphs.

The following three sections provide a brief description of some routing problems depending on whether the demand is on the links (arcs or edges), on the vertices, or even simultaneously on some vertices and some links of the graph.

## 3.1 Node Routing Problems

As mentioned above, the main characteristic of node routing problems is that the customers requiring service are represented as vertices of the graph. These problems

are among the most studied in Combinatorial Optimization and arise in many practical contexts such as freight distribution and collection, transportation, newspaper delivery, etc. Although there are many variants of these problems, they basically consist of minimizing the total distance traveled by one or more routes that visit all or some of the vertices of the graph. The following three problems are among the most important node routing problems and are considered the basis for studying related problems.

The **Traveling Salesman Problem (TSP)** was first mentioned in 1930 and it is one of the most studied optimization problems. TSP is an easy problem to explain but very difficult to solve. Given a road network, the problem consists of finding a minimum cost tour visiting all the cities exactly once. Usually, the TSP is defined in a complete graph and Karp (1972) proved that it is NP-*hard*. In-depth research has been carried out on TSP and many exact algorithms and heuristics have been developed, so that some instances with tens of thousands of cities can be solved to optimality and even problems with millions of cities can be approximated within a small fraction (1%).

The **Graphical Traveling Salesman Problem (GTSP)** is a variant of the TSP, introduced by Fleischmann (1985, 1988) and Cornuèjols et al. (1985), in which the graph does not necessarily to be complete and the vehicle is allowed to visit each vertex of the graph more than once.

The **Vehicle Routing Problem (VRP)** was first introduced by Dantzig and Ramser (1959) to model the problem in which a fleet of homogeneous trucks has to meet the oil demand of several gas stations from a central hub and with a minimum distance traveled. Five years later, Clarke and Wright (1964) generalized this problem to a linear optimization problem commonly found in the logistics and transportation domain. The VRP, as it is currently known, consists of, given a set of customers and a fleet of vehicles located in a central depot, finding a set of routes with minimum cost that, starting and ending in the depot, service the customers. The VRP is more difficult than TSP since it involves partitioning the set of customers so that they can be serviced by the vehicles and then defining the service order for each customer. Obviously, this problem is also NP-*hard*

## 3.2 Arc Routing Problems

The problems we deal with in this thesis are included in the class of Arc Routing Problems (ARPs). Unlike in Node Routing Problems, where the customers are represented by vertices in a graph, in ARPs the customers are represented by the arcs/edges of the

graph. ARPs consist of finding one or several routes, jointly traversing the arcs/edges requiring service and such that the total traveling cost is minimized. Within route planning, there are many traditional arc routing applications, such as garbage collection, road cleaning, mail delivery, electrical or rail networks inspection, snow removal, etc.

Although ARPs have been much less studied than node routing problems, the literature on arc routing problems is very extensive and impressive developments in the area have been achieved in the last 40 years. Around the 1980s, the articles by Assad et al. (1983), and Benavent et al. (1983) were among the first works to provide a general overview of the field. Bodin and Golden (1981) provide a detailed classification of these problems and Lenstra and Rinnooy-Kan (1981) discuss their complexity. Subsequently, and including new contributions, the works by Eiselt et al. (1995a,b), Assad and Golden (2000), and the book edited by Dror (2000) update much of the work done up to the year 2000 on arc routing problems. Corberán and Prins (2010) provided an annotated bibliography of more recent results. More lately, a comprehensive and up-to-date discussion of arc routing problems by renowned researchers is carried out in the book edited by Corberán and Laporte (2014). The article by Mourão and Pinto (2017) is a recent and exhaustive annotated bibliography. Finally, the article by Corberán et al. (2021) gives an updated vision of the current state of the art of ARPs and tries to foresee what will be the most relevant topics and research lines in this area in the next years.

As said in the Introduction, it was not until the 1960s that a Chinese mathematician, Meigu Guan (Mei-Ko Kwan) proposed in Guan (1962) what is known as the **Chinese Postman Problem (CPP)**. A postman is assigned a neighborhood in whose streets he must deliver the mail and his/her problem is to find the shortest route that, starting from the post office, traverses all the streets and returns to the office (see Figure 3.2).

Given an undirected connected graph $G = (V, E)$, the CPP can be posed as the problem of finding a tour traversing each edge of $E$ at least once with minimum total distance. In Edmonds (1965), it is proved that (the undirected version of) this problem is polynomially solvable by providing a polynomial algorithm which solves it to optimality. As the full description of the associated polyhedron is also known, the CPP is considered a "solved" problem and belongs to the complexity class $P$.

Afterwards, Edmonds and Johnson (1973) demonstrate that the directed version of the Chinese postman problem (DCPP) can also be solved in polynomial time. The authors showed that it is possible to solve efficiently the problem by solving a *transportation*

*problem*[1] to decide which arcs must be added to obtain a closed tour of minimum distance that traverses at least once each arc in the graph. Therefore, the DCPP is also considered a "solved" problem, and the full description of its associated polyhedron is known. The DCPP was introduced in a similar way in Orloff (1974) and Beltrami and Bodin (1974).

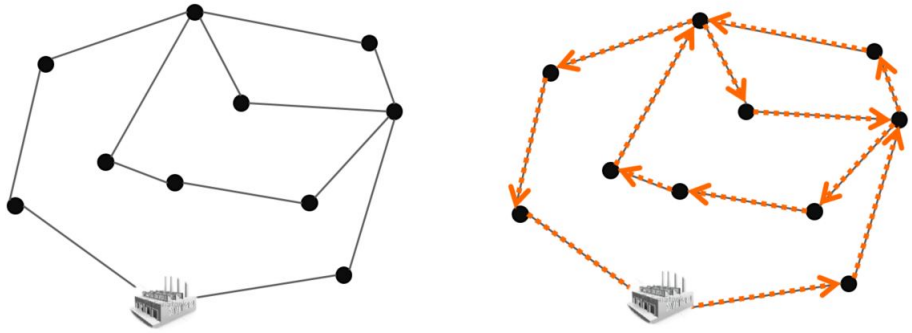


Figure 3.2: Chinese Postman Problem (CPP)

The CPP defined on a mixed graph (MCPP) is a much more difficult problem. Papadimitriou (1976) showed that, unlike the directed and undirected versions of the CPP, the MCPP is an NP-*hard* problem. Anyway, the study of the problem has provided a great deal of knowledge. For example, an optimal algorithm for the mixed CPP, which includes a polynomial-time procedure to identify a violated balanced-set inequality in a mixed graph if one exists, is presented in Nobert and Picard (1996). In Edmonds and Johnson (1973), it is proved that if $G$ is an even graph, the MCPP can be solved in polynomial time.

If we assume that the edges of the graph can be traversed in either direction but with different costs, depending on the direction of the traversal, we have the Windy Chinese Postman Problem (WPP). This problem, proposed by Guan (1984), contains as special cases the three previous ones and, therefore, is NP-*hard*. A solid study on the WPP can be found in the thesis of Win (1987), where it is shown that the WPP can be solved in polynomial time under some assumptions, among them if $G$ is Eulerian.

A generalization of the CPP is the **Rural Postman Problem (RPP)**, which also has its origin in the delivery of mail, but, in this case, in rural areas. Consider a postman delivering the mail at several villages. The postman has to traverse some roads or streets (without delivering mail) to travel among the villages or neighbors.

[1]For each vertex $i \in V$, let $d_i$ be the number of arcs entering $i$ minus the number of arcs leaving $i$. Let $S$ ($T$) be the set of vertices $i$ with $d_i > 0$ ($d_i < 0$). The DCPP can be formulated as the problem of finding the minimum cost transportation plan between the plants $S$ and the destinations $T$.

These "links", traversed (without being serviced) to move from one service area to another, are called *deadheading links*. The main difference between the CPP and the RPP is that in the former the set of links to be serviced is the set of links of the graph, while in the RPP there are links that are not necessary to service (see Figure 3.3).

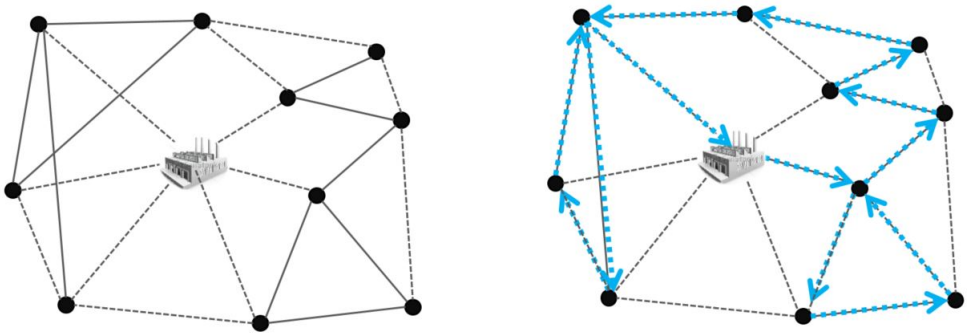


Figure 3.3: Rural Postman Problem (RPP)

Consider an undirected connected graph $G = (V, E)$ and let $E_R \subseteq E$ be a non-empty set whose edges are called *required edges*. The RPP is the problem of finding a tour in $G$ traversing each edge in $E_R$ at least once with minimum total cost. This problem was first proposed by Orloff (1974), and Lenstra and Rinnooy-Kan (1976) proved that it is NP-*hard*. Orloff pointed out that the complexity of RPP seems to increase with the number of connected components of the subgraph induced by the required edges. This was consistent with the result published in Frederickson (1979), where an exact recursive algorithm for the RPP, exponential in the number of such components, is proposed.

The versions of the RPP defined on directed (DRPP), mixed (MRPP), and windy graphs (WRPP), have also been studied. As with the undirected RPP, these three versions are NP-*hard*. For the DRPP, MRPP, and WRPP, an extensive polyhedral study has been done. Moreover, exact and approximate algorithms have been designed and implemented for the solution of these three problems.

Another ARP that has been widely studied is the well known **Capacitated Arc Routing Problem (CARP)**. The CARP was introduced in Golden and Wong (1981). It is a generalization of the RPP in which a demand is associated with each required edge or arc. In the CARP, the goal is to find a set of routes for a fleet of vehicles with limited capacities based at a depot, so that the total demand serviced on each route does not exceed the vehicle's capacity and the total cost is minimized (see Figure 3.4). The CARP has been extensively studied and many exact and heuristic algorithms have been proposed to solve it. The recent work by Pecin and Uchoa (2019) analyzes the best known exact algorithms for the CARP and proposes a new

branch-and-cut-and-price algorithm that solves almost all instances from the classical benchmark CARP sets.

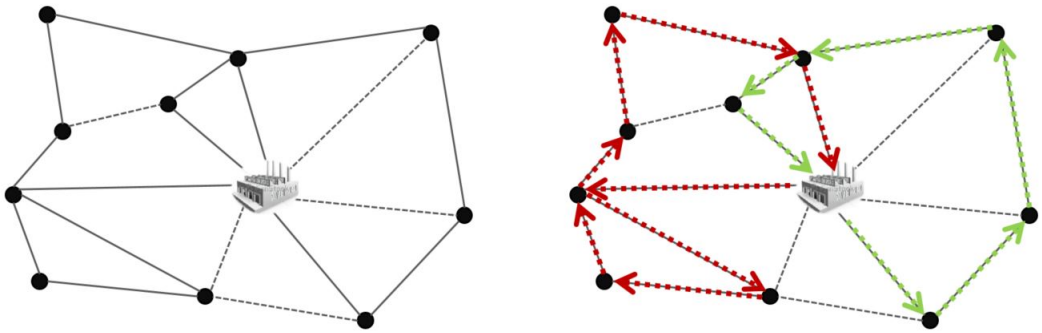


Figure 3.4: Capacitated Arc Routing Problem (CAPP)

During the last 40 years, several research groups around the world have been studying many arc routing problems, new ones and generalizations of others already studied, that allow modeling the new situations that real life presents. In parallel, there have been important developments in the design and implementation of exact algorithms, based on branch-and-cut, column generation and their hybridization, and sophisticated metaheuristic algorithms.

## 3.3 General Routing Problems

In arc routing problems the service demand is located on the arcs or/and edges of a network. However, there are real-life situations where the demand is also located at some vertices of the graph. The General Routing Problem (GRP) was first introduced by Orloff (1974). The GRP in an undirected graph $G = (V, E)$ consists of, given a subset of *required edges* $E_R \subseteq E$ and a subset of *required vertices* $V_R \subseteq V$, to find a tour with minimum total cost in $G$ that traverses each required edge and visits each required vertex at least once. Since the GRP contains the RPP as a special case, it is also an NP-*hard* problem (Lenstra and Rinnooy-Kan (1976)).

One of the reasons for studying the GRP lies in the number of problems that it contains as special cases. For example, if all edges in the graph are required, $E_R = E$, we have the CPP. If there are no required vertices $V_R = \emptyset$, the RRP results, and, if there are no required edges and all vertices in the graph are required, $E_R = \emptyset$ and $V_R = V$, the GRP reduces to a pure node routing problem, the GTSP. Hence, an ILP formulation for the GRP is also an RPP formulation. Thus, all inequalities valid for the RPP are valid for the GRP, and vice versa. Note that the algorithms developed for solving the RPP are also applicable for solving the GRP (see Corberán et al. (2001)).

The GRP versions defined for directed (DGRP), mixed (MGRP), and windy graphs (WGRP), have also been studied. In all of them it has been proved that they are NP-*hard*. In fact, as we will see in this thesis, we use the polyhedral study for the DGRP to support the polyhedral study for other more complex ARPs.

# Chapter 4

# Close-Enough Routing Problems

## 4.1 Introduction

In all the mentioned routing problems, the service is performed while the vehicle traverses a vertex or an arc/edge of the graph. These problems are found in many real-world applications such as meter reading, postal delivery or waste collection, where customers are located in streets or roads and service is performed while traversing them. In recent years, the development of new technologies has resulted in situations where it is not necessary for the vehicle to reach the exact position of the customer in order to perform its service, but simply to get close to the customer. This scenario is referred to as "close-enough" in routing problems, since the vehicle only has to pass close enough to the customer location. In Close-Enough Routing Problems (CERPs) we can distinguish between Close-Enough Traveling Salesman Problems (CETSPs), where customers are represented as nodes (they do not necessarily match the vertices of the graph), and Close-Enough Arc Routing Problems (CEARPs), where customers are represented as arcs or edges. Several variants of these problems have been extensively studied in the literature.

One of the most direct applications of the CERPs can be found in meter reading (see Figure 4.1). Technological innovations such as radio frequency identification (RFID) make it possible to remotely collect consumption data from gas, electricity, or water meters, instead of having to do it door-to-door as was customary years ago (Uribe-Pérez et al. (2016)). The meter sends a signal that describes the consumption of gas, electricity, or water, which is captured by the receiver if it is less than a certain distance. Thus, given a road or street network where the meters are located, there are vehicles with a radio frequency receiver that can read the consumption just getting

close to each meter. In this case, the vehicle/operator only has to enter the meter's coverage area to perform the service, without the need to physically visit all the customers, which saves time and money. The work by Eglese et al. (2014) provides an interesting summary of the models and methods proposed since the late 1970s in meter reading.

Figure 4.1: CERP application in meter reading

Another application of CERPs can be found in inventory management in large companies, especially those where there are many products and therefore checking them one by one is very time-consuming. RFID tags have revolutionized supply chain management by enabling warehouse managers to record inventory much more efficiently than they could by reading box numbers and manually recording them. But the scale of modern retail operations makes RFID scanning inefficient. Even with RFID technology, it can take a single large retail store a long time to perform a complete inventory review, which means that mismatches often go undiscovered until exposed by a customer request. MIT researchers have developed a system that enables aerial drones to read RFID tags from tens of meters away and identify the location of the tags with an average error of about 19 centimeters (see Figure 4.2). Therefore, to perform the inventory the drone does not need to traverse all the aisles of the warehouse for data collection. The researchers anticipate that the system could be used in large warehouses both for continuous monitoring to avoid inventory mismatches and for locating individual items so that employees can quickly and reliably respond to customer requests.

Drones with RFID receivers or built-in cameras (Figure 4.3) are identified by Aráoz et al. (2017) as the most suitable devices to perform certain tasks such as maintenance or surveillance quality control for networks maintenance and surveillance tasks. The drones do not have to fly over the nodes or lines to be monitored, but only to approach the target at a certain distance. Aráoz et al. indicated that in quality control for networks maintenance only a small subset of the edges of a network has to be traversed. They also argued that the CEARP is the most appropriate problem for modeling

Figure 4.2: Application CERP in Inventory Review

location/arc routing problems in which facilities have to be located at some given areas and connected among them by means of a route.

Figure 4.3: Application CERP in surveillance tasks

Yuan et al. (2007) and Behdani and Smith (2014) introduced another application in the robot monitoring of wireless sensor networks. As Yuan et al. (2007) point out, in a wireless sensor network, where sensors are geographically distant from each other, it may not be practical to require sensors to directly coordinate with each other to form a communication network due to the energy restriction. One possible solution is to employ a mobile robot, which can travel to all sensors, to download the data and finally return to its base station (starting position). Like in meter reading, the robot must be physically within its effective range.

This chapter addresses the CETSP and the CEARP as well as other variants studied in this thesis. Section 4.2 describes the nature of the CETSP and provides the literature review. The CEARP is described in more detail, as it is the main base of the problems we deal with in the thesis. Section 4.3 provides the literature review and the formal problem definition for the CEARP. Section 4.3.3 presents some of the variants of the original CEARP, focusing on the three problems studied in this thesis: the Profitable CEARP for a single vehicle, the Distance-Constrained CEARP, and the Min-Max CEARP.

## 4.2 Close-Enough Traveling Salesman Problems

Gulczynski et al. (2006) appear to be the first researchers to study the CETSP (with a single vehicle). In the CETSP, customers are located at points in the plane and the vehicle must travel within a required radius $r$ of each point to service the customer. In that paper, it is assumed that the vehicle is not restricted to a road network, that is, it can move between any pair of points in the plane following a straight line whose cost is the Euclidean distance. This situation would occur if, for example, the salesman is a drone and the service is performed by flying through a free flight zone in a neighborhood of each target point. The objective of the problem is to minimize the total distance traveled. The authors propose six heuristics to solve this problem under the assumption that all neighborhoods are discs of the same radius.

Since then, the CETSP and some variants in which the radius associated with each customer may vary, or the shape of the area around the customer is not a circle, have been studied by several authors. Dong et al. (2007) called this problem the automatic meter reading shortest tour problem (AMRSTP) and designed and implemented two heuristic approaches, a clustering-based algorithm and a convex hull-based algorithm. They also introduced a mixed integer non-linear programming formulation of the problem, but it was not specifically used in the design of the algorithm. In Yuan et al. (2007) an effective evolutionary approach was developed. It is able to find the shortest tour on all the benchmark instances, although with large computation time. Mennell (2009) proposed the Steiner-Zone Heuristic, an approximation algorithm based on the intersection of the neighborhoods. In the paper by Behdani and Smith (2014) was formulated a mixed-integer programming model based on a discretization scheme they used to define tighter lower and upper bounds. Coutinho et al. (2016) proposed an exact algorithm, based on branch-and-bound and second order cone programming. In Carrabs et al. (2017) a discretization scheme to compute both a lower and an upper bound was introduced, and a graph reduction algorithm to decrease the problem size was applied.

It was in Shuttleworth et al. (2008) where the CETSP was addressed over a realistic street network and modeled as a node routing problem on a directed graph. The authors proposed four heuristics to solve eight real-life instances with an average of 900 customers and 9000 streets each. These heuristics have essentially two phases: first a subset of streets to be traversed is selected, and then a route starting and ending at the depot and traversing all these streets is found. The first phase is obviously related to the resolution of a set covering problem, while the second consists of solving a directed rural postman problem (or a directed general routing problem if the depot is not incident with a selected street).

A closely related problem is the Covering Tour Problem (CTP) studied by Gendreau et al. (1997). The CTP is defined on an undirected graph $G = (V \cup W, E)$ where $W$ is a set of vertices that must be covered. The problem consists of determining a minimum length Hamiltonian cycle on a subset of $V$ such that every vertex of $W$ is within a prespecified distance from the cycle. For this problem, the authors presented an ILP formulation and several valid inequalities and propose a heuristic and a branch-and-cut algorithm. Baldacci et al. (2005) provided three scatter search methods for this problem. The multi-vehicle CTP has been studied in Hà et al. (2013) and Jozefowiez (2014). Finally, some special cases of CETSP were solved by polynomial-time approximation algorithms: the geometric covering salesman problem (GCSP) in Arkin and Hassin (1994), and the TSP with neighborhoods (TSPN) in Mata and Mitchell (1995) and in Dumitrescu and Mitchell (2003).

## 4.3 Close-Enough Arc Routing Problems

The Close-Enough Arc Routing Problem consists of finding a minimum cost route that starts and ends at the depot and traverses some of the streets of a network in such a way that all customers are serviced. A customer is serviced when the vehicle gets closer than a certain given distance. Therefore, it is assumed that we know which streets the vehicle can traverse to service each customer. The main characteristic of this problem is that the set of streets to be traversed is not known in advance, but rather is a decision variable.

### 4.3.1 Literature review

Drexl (2007) studied the problem without considering the presence of a depot. Instead of referring to the problem as CEARP, he referred to it as the Generalized Directed Rural Postman Problem. He noted that when each customer is serviced from a single arc, the problem reduces to the well known Directed Rural Postman Problem (DRPP)

and, therefore, the CEARP is *NP-hard*. He proposed a formulation and a branch-and-cut algorithm producing good computational results. A more recent version of his work was published in Drexl (2014). Hà et al. (2014) introduced the name Close-Enough Arc Routing Problem and proposed a new formulation, which they compared with that in Hà et al. (2012) and the one by Drexl (2007, 2014). Moreover, they presented a branch-and-cut algorithm that provided very good computational results on large-size instances.

More recently, Ávila et al. (2016b) presented two new mathematical formulations for the problem and studied the polyhedron of solutions associated with one of them. The authors also describe several new families of valid inequalities that they use to implement a new branch-and-cut algorithm. As a peculiarity, in that paper it is shown that the CEARP can be considered as a combination of two optimization problems, the Set Covering Problem (SCP) and the Directed General Routing Problem (DGRP). The authors compared the performance of their branch-and-cut algorithm with that published in Hà et al. (2014). In Cerrone et al. (2017) a new flow-based formulation was proposed, as well as some techniques to reduce the size of the graph. Using this new formulation, the results obtained in one of the sets of instances proposed in Hà et al. (2012) improved those of Hà et al., but were slightly worse than those of Ávila et al. (2016b). Therefore, as far as we know, the algorithm by Ávila et al. (2016b) is the best exact method to solve CEARP.

### 4.3.2 CEARP definition and formulation

We here present the definition of CEARP, together with a formulation and a set of valid inequalities to strengthen it, which are based on those proposed by Ávila et al. (2016b). Consider a strongly connected and directed graph $G = (V, A)$, with set of vertices $V$ and set of arcs $A$, and let $d_{ij} \geq 0$ be the distance/cost associated with the traversal of arc $(i, j) \in A$. Each costumer can be serviced whenever the vehicle traverses any arc among those located at a distance less than or equal to $r$ from the customer. Thus, a customer can also be defined/represented by that set of arcs. Hence, given a set of customers $\mathbb{H} = \{1, \ldots, L\}$, each customer $c \in \mathbb{H}$ has an associated set of arcs $H_c \subseteq A$ from which it can be serviced. These subsets $H_c$ do not need to be disjoint nor induce connected subgraphs. See Figure 4.4.

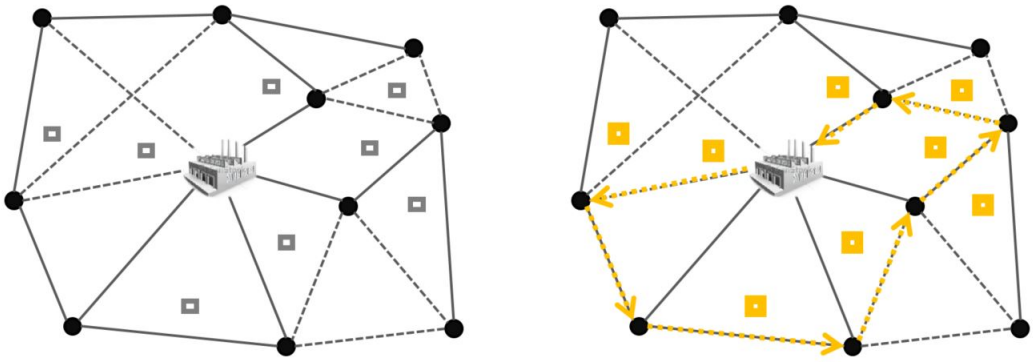


Figure 4.4: Close-Enough Arc Routing Problem (CEARP)

The notation we use for this problem is basically the same that we will use to formulate the other problems throughout this thesis. The set $H_1 \cup H_2 \ldots\ldots \cup H_L$ is denoted by $A_R$, called set of required arcs, and represents those arcs from whose traversal customers are serviced. Given a subset of vertices $S \subset V$, we define

- $\delta^+(S) = \{(i,j) \in A \ : \ \ i \in S, \ \ j \in V \setminus S\}$,
- $\delta^-(S) = \{(i,j) \in A \ : \ \ i \in V \setminus S, \ \ j \in S\}$,
- $\delta(S) = \delta^+(S) \cup \delta^-(S)$,
- $A(S) = \{(i,j) \in A \ : \ i,j \in S\}$,
- $A_R(S) = \{(i,j) \in A_R \ : \ i,j \in S\}$.

As usual, we will use $\delta^+(i)$ instead of $\delta^+(\{i\})$.

The Close-Enough Arc Routing Problem can be formally defined as the problem of finding a minimum cost tour, starting and ending at the depot (vertex 1), that services all the customers. Note that the tour must traverse at least one arc from each subset $H_c$, $c = 1, \ldots, L$ to service all the customers. In order to formulate the problem, we represent a CEARP solution using two sets of variables:

$x_{ij}$ = number of times arc $(i,j) \in A$ is traversed

$$y_{ij} = \begin{cases} 1, & \text{if at least one customer is serviced from the arc } (i,j) \in A_R, \\ 0, & \text{Otherwise.} \end{cases}$$

The CEARP can then be formulated as follows:

$$\text{Minimize} \sum_{(i,j)\in A} d_{ij}x_{ij}$$

s.t.:

$$x(\delta^+(i)) = x(\delta^-(i)) \qquad \forall\, i \in V \tag{4.1}$$

$$x(\delta^+(S)) \geq 1, \qquad \forall S \subseteq V \setminus \{1\} : \exists\, H_c \subseteq A(S) \cup \delta(S) \tag{4.2}$$

$$x(\delta^+(S)) \geq y_{ij}, \qquad \forall S \subset V \setminus \{1\} :$$
$$\nexists\, H_c \subseteq A(S) \cup \delta(S),\ \forall (i,j) \in A_R(S) \tag{4.3}$$

$$\sum_{(i,j)\in H_c} y_{ij} \geq 1, \qquad \forall c \in \mathbb{H} \tag{4.4}$$

$$x_{ij} \geq y_{ij}, \qquad \forall (i,j) \in A_R \tag{4.5}$$

$$x_{ij} \geq 0 \text{ and integer} \qquad \forall (i,j) \in A \tag{4.6}$$

$$1 \geq y_{ij} \geq 0 \text{ and integer} \qquad \forall (i,j) \in A_R, \tag{4.7}$$

Equations (4.1) ensure the symmetry on the vertices. The connectivity of the route is guaranteed by constraints (4.2) and (4.3). Inequalities (4.4) imply that, if a customer $c$ is serviced, at least an arc in $H_c$ is traversed. Constraints (4.6) and (4.7) are the nonnegativity and integrality inequalities.

In Ávila et al. (2016b), a CEARP tour is defined as any vector $(x, y) \in \mathbb{R}^{|\mathbb{A}|+|\mathbb{A}_\mathbb{R}|}$ satisfying constraints (4.1) to (4.7). The authors realized that any CEARP solution is a CEARP tour, but not vice versa. It should be noted that the above formulation allows subtours that are disconnected from the depot and do not service any customer, so that not all CEARP tours are CEARP solutions. However, since all distances $d_{ij}$ are nonnegative, there will always be an optimal CEARP tour that does not contain such subtours. In addition to the formulation, the authors also carried out a polyhedral study of the problem and obtained several valid inequalities and facets that helped to better understand the problem and its solution. Here we present them briefly and refer the reader to Ávila et al. (2016b) for a more detailed description.

**Connectivity inequalities.** Valid inequalities (4.8) are a generalization of the constraints (4.2). These new valid inequalities induce facets of the CEARP polyhedron if some conditions are satisfied.

$$x(\delta^+(S)) \geq 1 - y\big(H_c \cap A_R(V \setminus S)\big), \quad \forall S \subset V \setminus \{1\} \quad \forall\, c \in \mathbb{H} \tag{4.8}$$

**Parity inequalities.** Let $S \subset V$, $F \subseteq \delta_R(S)$, and a set of customers $\{c_1, c_2, ..., c_q\}$, with $|F| + q$ odd and $q > 0$, such that $H_{c_i} \cap \delta(S) \neq \emptyset$, $H_{c_i} \cap H_{c_j} \cap \delta(S) = \emptyset$, and $H_{c_i} \cap F = \emptyset$ for all $c_i, c_j,\ i, j = 1, \ldots, q$. Inequalities (4.9) are valid for CEARP.

$$x(\delta(S)) \geq 2y(F) - |F| + \sum_{i=1}^{q} \big(1 - 2y(H_{n_i} \backslash \delta(S))\big) + 1 \tag{4.9}$$

**K-C inequalities.** Consider a partition of the vertex set $V$ into $K$+1 subsets $\{M_0 \cup M_K, M_1, \ldots, M_{K-1}\}$ with $K \geq 3$. Let $\{I_1, I_2\}$ be a partition of the set $\{1, 2, \ldots, K-1\}$ such that for each $j \in I_1$, either $1 \in M_j$ or there is a $c_j$ such that $H_{c_j} \cap A_R(M_j) \neq \emptyset$, for each $j \in I_2$, there is a required arc $a_j \in A_R(M_j)$, and the induced subgraphs $G(M_i)$, $\forall i = 0, \ldots, K$ are strongly connected. For simplicity, we denote $A_R^{OK} = (M_0 : M_K)_R \cup (M_K : M_0)_R$. Let $F \subseteq A_R^{OK}$ be a set of arcs and let $\{c_1, c_2, \ldots, c_q\}$ a set of customers satisfying:

- $|F| + q$ is even,
- $H_{c_i} \cap A_R^{OK} \neq \emptyset \quad \forall i$,
- $H_{c_i} \cap H_{c_j} \cap A_R^{OK} = \emptyset \quad \forall i,\ \forall j$, and
- $H_{c_i} \cap F = \emptyset \quad \forall i$.

If we assume that $1 \in M_0$, the following inequality is valid for CEARP:

$$\begin{aligned}
&(K-2)\Big(x(M_0 : M_K) + x(M_K : M_0)\Big) + \sum_{\substack{0 \leq i,j \leq K \\ (i,j) \neq (0,K)}} |i-j| x(M_i : M_j) \\
&\geq (K-2)\big(2y(F) - |F|\big) + (K-2) \sum_{i=1}^{q} \big(1 - 2y(H_{c_i} \setminus A_R^{OK})\big) \\
&+2 \sum_{j \in I_1} \big(1 - y(H_{c_j} \setminus A(M_j))\big) + 2 \sum_{j \in I_2} y_{a_j}
\end{aligned} \tag{4.10}$$

**Dominance inequalities.** This last class of inequalities was previously introduced by Hà et al. (2014). Given two arcs $(i_1, j_1),\ (i_2, j_2) \in A_R$, the first dominates the second if for any customer $c \in \mathbb{H}$ such that $(i_2, j_2) \in H_c$ then $(i_1, j_1) \in H_c$. For any pair of arcs such that $(i_1, j_1)$ dominates $(i_2, j_2)$, or vice versa, the inequalities are defined as

$$y_{i_1 j_1} + y_{i_2 j_2} \leq 1. \tag{4.11}$$

Ávila et al. (2016b) also implemented a branch-and-cut algorithm based on this formulation, which incorporates separation algorithms for the valid inequalities they proposed. They carried out an extensive computational analysis over various sets of

CEARP instances and the results show that this algorithm outperforms the other existing exact methods.

### 4.3.3 CEARP variants

Many papers can be found in the literature that address variants of the original CEARP. Three of these variants are the problems studied in this thesis and described below: the Profitable CEARP for a single vehicle, and the Distance-Constrained CEARP and the Min-Max CEARP for multiple vehicles.

#### Profitable CEARP

The Profitable Close Enough Arc Routing Problem (PCEARP) is a new variant of the CEARP in which not all customers have to be serviced, but only those more interesting from the point of view of the profit provided. In this problem, a profit is associated with each customer and it is collected (only once) when the customer is serviced. The goal is to find a tour maximizing the difference between the total profit collected and the travel distance.

This problem is addressed in Chapter 5, where we present two different formulations and, for one of them, we carry out a study of its associated polyhedron and introduce several families of valid inequalities. We propose a heuristic that provides good feasible solutions in short times and whose lower bounds are very useful in a sophisticated branch-and-cut method that we have also implemented. An extensive computational analysis is performed on several sets of instances generated specifically for this problem.

#### Distance-Constrained CEARP

The CEARP is defined for a single vehicle but, in practical applications where the number of customers is very high and a single vehicle cannot perform all services, the service must be performed by a fleet of vehicles (or one vehicle performing several routes). In this case, the aim is to balance the routes. Even though there are many possibilities to do this, we focus on those that are related to real-world applications of the problem. In CEARP applications, such as meter reading, inventory taking, or wireless sensor network monitoring, it would make sense to consider a maximum capacity for the devices performing the task. In all these cases, due to the autonomy of vehicles (or drones or mobile robots), what must be considered is a time, or maximum length, that each vehicle can use.

In the distance-constrained CEARP (DC-CEARP) a CEARP is studied in which the length or maximum time of each route is limited and the objective is to minimize the total length. Given a homogeneous fleet of vehicles, the DC-CEARP consists of finding a set of routes with total minimum cost, that start and end at a depot, service all the customers, and such that the length of each route does not exceed a certain limit. In Ávila et al. (2017), the authors described the problem, proposed four different formulations using different types of variables, and presented some valid inequalities. To compare on them, they designed and implemented four branch-and-cut algorithms for its solution, and extensive computational experiments over various sets of instances comparing the performance of the different algorithms were provided. The results show that two of the proposed algorithms have a significantly better performance than the others and are therefore the most promising. They noted that the problem is NP-*hard*, since it generalizes the CEARP.

Two chapters of this thesis are dedicated to this variant of CEARP. In Chapter 6, we describe an approximate algorithm for the DC-CEARP. We propose a multi-start matheuristic that incorporates an effective branch-and-cut method for the CEARP in order to optimize the routes obtained. The results achieved with two versions of the matheuristic are compared to those obtained with the branch-and-cut in Ávila et al. (2017). In Chapter 7, we show a new formulation for the DC-CEARP that combines the best features of the previously existing ones. Moreover, an exhaustive study of its associated polyhedron is presented, as well as several families of valid inequalities. Based on this new formulation, we have also designed and implemented a new branch-and-cut algorithm that uses an upper bound obtained by the matheuristic algorithm. An extended computational analysis has been performed in which the efficiency of the branch-and-cut algorithm is tested.

### Min-Max CEARP

Min-max objectives are quite common in multi-vehicle routing problems because minimizing the length of the longest route tends to balance the length or cost of the planned routes. Moreover, if the travel times are proportional to the travel distance, the last customer serviced is serviced as early as possible. The Min-Max Close-Enough Arc Routing Problem (MM-CEARP) is another variant of CEARP in which a fleet of homogeneous vehicles must service a set of customers in such a way that the lengths of the routes are balanced. The MM-CEARP consists of finding a set of routes for the vehicles, all of them starting and ending at the depot, jointly servicing all the customers, and such that the length of the longest route is minimized.

In chapter 8, we define this variant of CEARP and focus on its formulation and solution. We propose two different formulations for the MM-CEARP, one based on routing and servicing variables, and other based on a set covering formulation. We devise and implement two exact methods for solving the problem optimally, a branch and cut and a branch-and-price algorithm. To provide exact algorithms with an upper bound, we have implemented a two-step heuristic algorithm that includes a constructive phase and a local search. All the algorithms are compared in benchmark instances through extensive computational experiments.

**Other related problems**

A further variant of CEARP arises when there is uncertainty in the service provided to customers. For example, in meter reading, the stochasticity lies in the uncertainty of collecting data due to failed transmissions. For this reason, the Stochastic CEARP (SCEARP) was studied in Renaud et al. (2017). Authors assume that it is possible that the remote reading of the meter fails. Then, they introduce the probability of reading a meter as a function of the distance of the customer from the route taken by the operator. For this problem, authors propose a mathematical formulation, give some preprocessing properties, and propose a cutting-plane algorithm and several heuristics for its solution.

Aráoz et al. (2017) studied the Generalized Arc Routing Problem (GARP) on an undirected graph. This problem is a special case of the CEARP in which the customers are associated with clusters of edges that are pairwise-disjoint connected subgraphs. The authors present some facets and valid inequalities for the GARP and propose a branch-and-cut algorithm to solve it. Note that GARP can be seen as the arc routing counterpart of the Generalized Travelling Salesman Problem (GTSP) in the NRPs, in which the set of vertices of a given graph is partitioned into clusters and a route is sought that visits at least one vertex of each cluster.

A combination of a NRP and an ARP is provided in Russo et al. (2019). Both the CEARP and the CETSP are used to model routing of drones since in many real-life applications the drones are restricted to fly along the streets or moving corridors, and in certain areas they are free to move. In this work, the Mixed-Constrained Routing Problem (MCRP) is proposed, which distinguishes between zones where the flight is free (CETSP), where drones can fly freely between any two points in the plane, and zones where drones movement is restricted to certain flight corridors (CEARP). The authors proposed two possible approaches to face this scenario, a heuristic algorithm, based on a local search procedure, and a genetic algorithm.

# Chapter 5

# The Profitable Close-Enough Arc Routing Problem

Routing problems with profits deal with situations where the customers to be serviced must be chosen from a set of potential customers that have an associated profit that is collected when they are serviced. They basically consist of designing one or more routes that service the chosen customers and in such a way as to optimize an objective defined as a function of the cost or/and the profit. A routing problem with profits is called *profitable* when its objective is to maximize the difference between the profit collected and the cost of the routes (Feillet et al. (2005)).

Node routing problems with profits (NRPPs) have been addressed in many articles lately. We refer the reader to the surveys by Vansteenwegen et al. (2011) and Archetti et al. (2014b). Similarly to NRPPs, arc routing problems with profits (ARPPs) are a growing area in which a large number of papers have been devoted to studying many of these problems. Malandraki and Daskin (1993) is the first paper dealing with an ARPP. In that paper, the authors introduce the Maximum Benefit Chinese Postman Problem on a directed graph, where a profit, which can be different in each traversal, is collected each time an arc is traversed with service, and the goal is to obtain a tour that maximizes the profit. A more appropriate name for this problem, the Profitable Rural Postman Problem with Multiple Visits (PRPPMV), was suggested by Archetti and Speranza (2014). Other papers on the undirected or directed PRPPMV are Pearn and Wang (2003) and Pearn and Chiu (2005), where several heuristics are proposed, and Corberán et al. (2013), where the polyhedron associated with the problem is studied and an exact algorithm is proposed.

The Profitable Rural Postman Problem (PRPP), in which there is a profit associated just to the edges of a given subset, was proposed in Aráoz et al. (2006), and an exact method is described in Aráoz et al. (2009). Note that the PRPP, also called Prize-Collecting Arc Routing Problem and Privatized Rural Postman Problem, is a special case of the PRPPMV where there is a profit associated with some edges, while the remaining ones have 0 profit. In Archetti et al. (2014a) and Colombi and Mansini (2014), the PRPP defined on a directed graph is considered, while the version defined on a windy graph has been studied in Ávila et al. (2016a). The Profitable Capacitated Arc Routing and other related problems are addressed in Benavent et al. (2015).

In all the problems mentioned above, the service is assumed to be performed while the vehicle traverses an arc or edge of the graph. In this thesis, we introduce a new variant of a CEARP in which not all customers have to be serviced, but only those more interesting from the point of view of the profit provided. This problem, which we call the Profitable Close Enough Arc Routing Problem (PCEARP), can also be seen as a Profitable Rural Postman Problem in which the required arcs are grouped into families associated with customers, and to service them it is enough to traverse one of their associated arcs. As the PCEARP generalizes both the CEARP (when the profits associated with all the customers are very large) and the PRPP (when each required arc defines a customer), it is an NP-*hard* problem covering different practical applications.

The contribution of this chapter is fourfold. First, we introduce a new problem that combines the most interesting features of two existing NP-*hard* routing problems: the possibility to select the customers to service on the basis of the net profit associated with them, and the possibility to provide the service by traversing one of the streets that are close enough to them. Second, we present a formulation for the PCEARP for which a study of its associated polyhedron is performed, and introduce several families of valid inequalities. Third, we propose a simple but fast heuristic providing good feasible solutions in short times and, fourth, we present a sophisticated branch-and-cut method that is able to optimally solve instances up to 600 customers, 300 vertices, and 1500 arcs within the time limit of one hour.

The chapter is organized as follows. In Section 5.1 we define the problem and present the proposed formulation. The PCEARP polyhedron is studied in Section 5.2. Section 5.3 is devoted to the description of the proposed heuristic algorithm. The separation procedures for the identification of violated inequalities of the types described, as well as the branch-and-cut algorithm, are presented in Section 5.4. Section 5.5 describes the generated instances and the computational results obtained with the heuristic and the branch-and-cut algorithm. Finally, some conclusions are given in Section 5.6.

## 5.1 Problem definition and formulation

Let $G = (V, A)$ be a directed and strongly connected graph with set of vertices $V$ (vertex 1 denotes the depot) and set of arcs $A$. For each arc $(i, j) \in A$, let $d_{ij} \geq 0$ be its associated length/distance. Then, let $\mathbb{H}$ represents a set of customers. Each customer $c \in \mathbb{H}$ is associated with a set of arcs $H_c \subseteq A$ and is serviced when at least one of the arcs in $H_c$ is traversed. Associated with each customer $c \in \mathbb{H}$ there is a profit $p_c \geq 0$ that is collected (only once) if the customer is serviced. Note that the subsets $H_c$ do not need to be disjoint nor the corresponding induced subgraphs must be connected. The Profitable Close-Enough Arc Routing Problem (PCEARP) consists of finding a tour (closed walk) on $G$, which starts and ends at the depot, maximizing the difference between the sum of the profits collected and the total length of the tour.

In this chapter, we use a similar notation to that presented for the CEARP in the previous one. The arcs in the set $A_R = H_1 \cup H_2 \cup \ldots \cup H_L$ are called required arcs, while non-required arcs are those in the set $A_{NR} = A \setminus A_R$. Given two sets $S, T \subset V$, we define $(S : T) = \{(i, j) \in A : i \in S,\ j \in T\}$ and $(S, T) = (S : T) \cup (T : S)$. In particular, $\delta^+(S) = (S : V \setminus S)$, $\delta^-(S) = (V \setminus S : S)$ and $\delta(S) = (S, V \setminus S)$, and $A(S) = (S : S)$. If, for any of the previous subsets, we want to refer only to its required arcs, we will use the notation $(S, T)_R$, $\delta_R^+(S)$, $\delta_R^-(S)$, and $A_R(S)$. Finally, given a vector $x$ indexed on the set of arcs and given a set $F \subseteq A$, $x(F) = \sum_{(i,j)\in F} x_{ij}$.

A tour for the PCEARP is a closed walk on $G$, starting and ending at the depot. Each tour on $G$ can be represented by an integer vector $x = (x_{ij}) \in \mathbb{Z}^{|A|}$ where, for each arc $(i, j) \in A$, $x_{ij}$ is the number of times the vehicle traverses $(i, j)$. Although vector $x$ is enough to know which customers are serviced (those $c$ such that $x_{ij} > 0$ for any $(i, j) \in H_c$), we consider additional variables in order to take into account, only once, the profit $p_c$ collected when a customer $c$ is serviced. For each customer $c \in \mathbb{H}$, a binary variable $z_c$ is defined taking value 1 if the customer $c$ is serviced and 0 otherwise. We want to point out that, since we have assumed that profits $p_c$ are not negative, all customers $c$ for which an arc in $H_c$ is traversed are worth being serviced and the optimal solutions of the problem will always satisfy this. However, we will also consider as feasible those solutions where a customer $c$ is not serviced even if some of the arcs in $H_c$ are traversed. This makes the polyhedral study in Section 5.2 easier. Thus, a tour for the PCEARP is represented by an integer vector $(x, z) \in \mathbb{Z}^{|A|+|\mathbb{H}|}$, with variables defined as:

$x_{ij}$ = number of times arc $(i, j) \in A$ is traversed, and

$$z_c = \begin{cases} 1, & \text{if the customer } c \text{ is serviced,} \\ 0, & \text{otherwise.} \end{cases}$$

The PCEARP can be formulated as

$$\text{Maximize} \sum_{c\in\mathbb{H}} p_c z_c - \sum_{(i,j)\in A} d_{ij}x_{ij}$$

s.t.:

$$x(\delta^+(i)) = x(\delta^-(i)) \qquad \forall\, i \in V \tag{5.1}$$

$$x(\delta^+(S)) \geq z_c - x(H_c \cap A(V \setminus S)) \qquad \forall S \subset V \setminus \{1\},\ \forall c \in \mathbb{H} \tag{5.2}$$

$$x(H_c) \geq z_c \qquad \forall c \in \mathbb{H} \tag{5.3}$$

$$x_{ij} \geq 0 \ \text{ and integer} \qquad \forall (i,j) \in A \tag{5.4}$$

$$1 \geq z_c \geq 0 \text{ and integer} \qquad \forall c \in \mathbb{H}, \tag{5.5}$$

Equations (5.1) are the symmetry conditions on the vertices. The connectivity of the routes is guaranteed by constraints (5.2). Note that, either if the vehicle does not service customer $c$, $z_c = 0$, or if it services $c$ and traverses an arc in $H_c \cap A(V \setminus S)$, $z_c - x(H_c \cap A(V \setminus S)) \leq 0$, and hence the inequality is satisfied. If the vehicle services customer $c$ without traversing any arc in $H_c \cap A(V \setminus S)$, it must cross the cutset $\delta(S)$ in order to traverse an arc in $H_c \cap A(S)$. Inequalities (5.3) imply that, if a customer $c$ is serviced, at least an arc in $H_c$ is traversed. Finally, (5.4)–(5.5) define the domain of the variables.

Note that this is not a complete formulation for the PCEARP. Although each tour for the PCEARP $(x, z) \in \mathbb{Z}^{|A|+|\mathbb{H}|}$ satisfies (5.1)–(5.5), there are vectors $(x, z) \in \mathbb{Z}^{|A|+|\mathbb{H}|}$ satisfying (5.1)–(5.5) that are not tours for the PCEARP because they can contain subtours disconnected from the depot. Nevertheless, since all the arc lengths $d_{ij}$ are nonnegative, there will always be an optimal PCEARP solution that does not contain any of these subtours.

## 5.2 The PCEARP polyhedron

Let us call *PCEARP tour* to any tour on $G$ starting and ending at the depot, and also to its corresponding integer vector $(x, z) \in \mathbb{Z}^{|A|+|\mathbb{H}|}$. Let PCEARP($G$) be the convex hull of all the PCEARP tours,

$$\text{PCEARP}(G) = conv\big\{(x, z) \in \mathbb{Z}^{|A|+|\mathbb{H}|} : (x, z) \text{ is a PCEARP tour}\big\}.$$

It can be seen that PCEARP($G$) is an unbounded polyhedron. Furthermore, constraints (5.1) to (5.5) from the formulation are valid for PCEARP($G$). We are going to study conditions under which the inequalities deduced from the formulation induce facets of PCEARP($G$). For this purpose, we need to know the dimension of PCEARP($G$).

**Theorem 12.** *If $G$ is a strongly connected graph, then* $\dim(\text{PCEARP}(G)) = |A| + |\mathbb{H}| - |V| + 1$.

*Proof.* Since $|V|-1$ of the $|V|$ symmetry equations (5.1) are linearly independent, we have that $\dim(\text{PCEARP}(G)) \leq |A| + |\mathbb{H}| - |V| + 1$.

Consider now the Directed General Routing Problem (DGRP) on $G$ with all the arcs in $A_R$ as required arcs and the depot as a required vertex. The DGRP is the problem that consists of finding a tour in $G$ with minimum cost traversing each required arc and visiting each required vertex at least once. Since $G$ is strongly connected, $\dim(\text{DGRP}(G)) = |A| - |V| + 1$ (see Ávila et al. (2015)), and there are $m+1$ affinely independent DGRP tours $x^1, x^2, \ldots, x^{m+1}$, where $m = |A| - |V| + 1$. Note that these tours traverse all the arcs in $A_R$ and visit the depot and, if we complete them with vectors $z \in \mathbb{Z}^{|\mathbb{H}|}$, we will obtain PCEARP tours $(x, z)$.

Let $z^0 \in \mathbb{Z}^{|\mathbb{H}|}$ the null vector, and $z^c$, $1 \leq c \leq |\mathbb{H}|$, the vector with all its entries equal to 0 except an 1 in position $c$, which represents the service of customer $H_c$. Obviously, $z^1, z^2, \ldots, z^{|\mathbb{H}|}$ are linearly independent vectors. Then $(x^1, z^0), \ldots, (x^{m+1}, z^0), (x^1, z^1), \ldots, (x^1, z^{|\mathbb{H}|})$ are $m+1+|\mathbb{H}|$ affinely independent PCEARP tours and $\dim(\text{PCEARP}(G)) \geq m + |\mathbb{H}| = |A| + |\mathbb{H}| - |V| + 1$. □

In the following, we will assume that $G$ is a strongly connected graph.

### 5.2.1 Facet-inducing inequalities obtained from the formulation

In this section we study conditions under which the inequalities in the formulation induce facets of PCEARP($G$).

**Theorem 13.** *Let $(i,j) \in A$ such that $G \setminus \{(i,j)\}$ is strongly connected. Inequality $x_{ij} \geq 0$ induces a facet of* PCEARP($G$) *if, and only if, $H_c \neq \{(i,j)\}$, for all $c \in \mathbb{H}$.*

*Proof.* If there is a customer $c$ such that $H_c = \{(i,j)\}$ then its associated inequality (5.3) is $x_{ij} \geq z_c$ and, therefore, $x_{ij} \geq 0$ cannot induce a facet. Otherwise, the PCEARP instance defined on graph $G \setminus \{(i,j)\}$ has $|A| - 1$ arcs and $|\mathbb{H}|$ customers. If graph

$G \setminus \{(i,j)\}$ is strongly connected, from Theorem 12 we obtain that dim(PCEARP($G \setminus \{(i,j)\}$))= $|A| - 1 + |\mathbb{H}| - |V| + 1 = m - 1$, where $m$ =dim(PCEARP($G$)), and we can find $(x,z)^1, \ldots, (x,z)^m$ affinely independent PCEARP tours on $G \setminus \{(i,j)\}$. After completing these tours with a zero component corresponding to the arc $(i,j)$, we obtain $m$ PCEARP tours on $G$ that are also affinely independent and satisfy $x_{ij} = 0$. Therefore $x_{ij} \geq 0$ induces a facet of PCEARP($G$). □

**Theorem 14.** *The inequality $z_c \geq 0$, for all $c \in \mathbb{H}$, induces a facet of* PCEARP($G$).

*Proof.* W.l.o.g. we will prove it for customer 1, i.e., for inequality $z_1 \geq 0$. Consider the Directed General Routing Problem (DGRP) on $G$ with all the arcs in $A_R$ as required arcs and the depot as a required vertex. Since $G$ is strongly connected, dim(DGRP($G$))=$|A| - |V| + 1 = m$ (Ávila et al. (2015)), and there are $m+1$ affinely independent DGRP tours $x^1, x^2, \ldots, x^{m+1}$. Note that these tours traverse all the arcs in $A_R$ and visit the depot. We complete them with vectors $z \in \mathbb{Z}^{|\mathbb{H}|}$ to obtain PCEARP tours $(x,z)$.

Let $z^{[c]}$ be the vector indexed in $\mathbb{H}$ with all its entries equal to zero except for $z_c = 1$, and let $z^{[0]}$ be the vector with all its entries equal to zero. The vectors $(x^1, z^{[0]}), \ldots, (x^{m+1}, z^{[0]}), (x^1, z^{[2]}), , \ldots, (x^1, z^{[|\mathbb{H}|]})$ are $m + |\mathbb{H}| = |A| + |\mathbb{H}| - |V| + 1 =$ dim(PCEARP($G$)) PCEARP tours satisfying $z_1 = 0$. They are affinely independent because, after subtracting the first one from all the other vectors we obtain $(x^2 - x^1, 0), \ldots, (x^{m+1} - x^1, 0), (0, z^{[2]}), \ldots, (0, z^{[|\mathbb{H}|]})$, which are linearly independent. Therefore, the inequality $z_1 \geq 0$ is facet-inducing for the PCEARP($G$). □

**Theorem 15.** *The inequality $z_c \leq 1$, for all $c \in \mathbb{H}$, induces a facet of* PCEARP($G$).

*Proof.* Again we will prove it for inequality $z_1 \leq 1$. Consider the DGRP on $G$ with all the arcs in $A_R$ as required arcs and the depot as a required vertex. Since $G$ is strongly connected, dim(DGRP($G$))=$|A| - |V| + 1 = m$, and there are $m+1$ affinely independent DGRP tours $x^1, x^2, \ldots, x^{m+1}$ traversing all the arcs in $A_R$ and visiting the depot. We complete them to obtain PCEARP tours $(x,z)$.

Let $z^{[*]} \in \mathbb{Z}^{|\mathbb{H}|}$ be the vector with all its entries equal to 1, and $z^{[\bar{c}]}$, $2 \leq c \leq |\mathbb{H}|$, the vector with all its entries equal to 1 except a 0 in position $c$. The vectors $(x^1, z^{[*]}), \ldots, (x^{m+1}, z^{[*]}), , (x^1, z^{[\bar{2}]}), \ldots, (x^1, z^{[|\bar{\mathbb{H}}|]})$ are PCEARP tours satisfying $z_1 = 1$. They are affinely independent vectors (note that $z^{[\bar{2}]}, \ldots, z^{[|\bar{\mathbb{H}}|]}$ are linearly independent vectors) and its number is $m + |\mathbb{H}| = |A| + |\mathbb{H}| - |V| + 1 =$ dim(PCEARP($G$)). Hence, the inequality $z_1 \leq 1$ is facet-inducing. □

**Theorem 16.** *Inequalities* (5.3)*,* $x(H_c) \geq z_c$ *for all* $c \in \mathbb{H}$*, induce a facet of* PCEARP$(G)$ *if the following conditions hold:*

**(a)** *Graph* $G \setminus H_c$ *is strongly connected.*

**(b)** *For each arc* $a \in H_c$*, there is a tour on* $G$ *that traverses arc* $a$ *but does not traverses any other arc in* $H_c \setminus \{a\}$*.*

*Proof.* We will do the proof also for customer 1, i.e., for inequality $z_1 \leq x(H_1)$. Let us consider the DGRP defined on graph $G \setminus H_1$ where all the arcs in $A_R \setminus H_1$ are taken as required and the depot is a required vertex. Given that $G \setminus H_1$ is a strongly connected graph (condition a), $\dim(\text{DGRP}(G \setminus H_1)) = |A| - |H_1| - |V| + 1 = m - |H_1|$, where $m = |A| - |V| + 1$, and there are $m - |H_1| + 1$ affinely independent DGRP tours in $G \setminus H_1$. We complete these tours with a zero component for each arc in $H_1$ and we obtain , $x^1, x^2, \ldots, x^{m-|H_1|+1}$ affinely independent DGRP tours in $G$ traversing all the arcs in $A_R \setminus H_1$ and not traversing the arcs in $H_1$.

Let us suppose first that there is no other customer set $H_c$ such that $H_c \subseteq H_1$. In this case, all the above DGRP tours traverse at least an arc in each $H_c$, for $c \neq 1$. Let $z^{[c]}$ be the vector indexed in $\mathbb{H}$ with all its entries equal to zero except for $z_c = 1$, and let $z^{[0]}$ be the vector with all its entries equal to zero. The following are $m - |H_1| + 1 + |\mathbb{H}| - 1$ PCEARP tours satisfying $z_1 = 0 = x(H_1)$:

$$(x^1, z^{[0]}), \ldots, (x^{m-|H_1|+1}, z^{[0]}), (x^1, z^{[2]}), \ldots, (x^1, z^{[|\mathbb{H}|]}).$$

Furthermore, from condition (b), for each arc $a \in H_1$, there is a tour on $G$ traversing $a$ but not traversing any other arc in $H_1 \setminus \{a\}$. Let $x^{[1]}, \ldots, x^{[|H_1|]}$ be such tours on $G$. The following $|H_1|$ PCEARP tours satisfy $z_1 = 1 = x(H_1)$ (recall that $z^{[1]}$ is the vector with all its entries equal to zero except for $z_1 = 1$):

$$(x^{[1]}, z^{[1]}), \ldots, (x^{[|H_1|]}, z^{[1]}).$$

Hence, we have $m - |H_1| + 1 + |\mathbb{H}| - 1 + |H_1| = m + |\mathbb{H}| = |A| + |\mathbb{H}| - |V| + 1$ PCEARP tours satisfying $z_1 = x(H_1)$. If we arrange them as the rows of a matrix we obtain the block matrix in Figure 5.1.a. After subtracting the first row from all the other rows and rearranging the blocks we obtain the block matrix in Figure 5.1.b, which has all its $|\mathbb{H}| + |A| - |V|$ rows linearly independent. Hence, the $|A| + |\mathbb{H}| - |V| + 1$ PCEARP tours satisfying $z_1 = x(H_1)$ are affinely independent and, therefore, inequality $z_1 \leq x(H_1)$ induces a facet of PCEARP$(G)$.

| $A \setminus H_1$ | $H_1$ | $z_1$ | $z_2 \ \dots \ z_{\lvert\mathbb{H}\rvert}$ |
|---|---|---|---|
| $x^1$ <br> $\vdots$ <br> $x^{m-\lvert\mathbb{H}\rvert+1}$ | 0 | 0 | 0 |
| $x^1$ <br> $\vdots$ <br> $x^1$ | 0 | 0 | I |
| $x^{[1]}$ <br> $\vdots$ <br> $x^{[\lvert H_1\rvert]}$ | I | 1 | 0 |

(a)

| $z_2 \ \dots \ z_{\lvert\mathbb{H}\rvert}$ | $H_1$ | $z_1$ | $A \setminus H_1$ |
|---|---|---|---|
| I | 0 | 0 | 0 |
| 0 | I | 1 | $x^{[1]} - x^1$ <br> $\vdots$ <br> $x^{[\lvert H_1\rvert]} - x^1$ |
| 0 | 0 | 0 | $x^2 - x^1$ <br> $\vdots$ <br> $x^{m-\lvert\mathbb{H}\rvert+1} - x^1$ |

(b)

| $A \setminus H_1$ | $a_1$ | $a_2$ | $a_3$ | $\cdots$ | $a_{\lvert H_1\rvert}$ | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $\cdots$ | $z_{\lvert\mathbb{H}\rvert}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $x^1$ <br> $\vdots$ <br> $x^{m-\lvert\mathbb{H}\rvert+1}$ | 0 | | | | | 0 | | | 0 | | |
| $x^1$ <br> $\vdots$ <br> $x^1$ | 0 | | | | | 0 | | | I | | |
| $x^{[2]}$ | 1 | 0 | 0 | $\cdots$ | 0 | 0 | 1 | 0 | 0 | $\cdots$ | 0 |
| $x^{[3]}$ | 0 | 1 | 0 | $\cdots$ | 0 | 0 | 0 | 1 | 0 | $\cdots$ | 0 |
| $x^{[1]}$ <br> $\vdots$ <br> $x^{[\lvert\mathbb{H}\rvert]}$ | I | | | | | 1 0 0 <br> $\vdots$ $\vdots$ $\vdots$ <br> 1 0 0 | | | 0 | | |

(c)

| $z_4 \ \cdots \ z_{\lvert\mathbb{H}\rvert}$ | $a_1$ | $a_2$ | $a_3$ | $\cdots$ | $a_{\lvert H_1\rvert}$ | $z_1$ | $z_2$ | $z_3$ | $A \setminus H_1$ |
|---|---|---|---|---|---|---|---|---|---|
| I | 0 | | | | | 0 | | | 0 |
| 0 | I | | | | | 1 0 0 <br> $\vdots$ $\vdots$ $\vdots$ <br> 1 0 0 | | | $x^{[1]} - x^1$ <br> $\vdots$ <br> $x^{[\lvert\mathbb{H}\rvert]} - x^1$ |
| 0 $\cdots$ 0 | 1 | 0 | 0 | $\cdots$ | 0 | 0 | 1 | 0 | $x^{[2]} - x^1$ |
| 0 $\cdots$ 0 | 0 | 1 | 0 | $\cdots$ | 0 | 0 | 0 | 1 | $x^{[3]} - x^1$ |
| 0 | 0 | | | | | 0 | | | $x^2 - x^1$ <br> $\vdots$ <br> $x^{m-\lvert\mathbb{H}\rvert+1} - x^1$ |

(d)

Figure 5.1: Matrices appearing in the proof of Theorem 16

Suppose now that there are other customer arc sets inside $H_1$. For the sake of simplicity, let us suppose that $H_2 \subseteq H_1$ and $H_3 \subseteq H_1$. We select two arcs $a_1 \in H_2 \cap H_1$ and $a_2 \in H_3 \cap H_1$. In this case, the above vectors $(x^1, z^{[2]})$ and $(x^1, z^{[3]})$ are not feasible PCEARP tours because, $x^1$ does not traverse any arc in $H_2$ or in $H_3$, while customers 2 and 3 are serviced ($z_2 = 1$ and $z_3 = 1$, respectively). We now replace these two PCEARP tours by the following ones. Let $x^{[1]}$ be the above tour on $G$ traversing $a_1$ but not traversing any other arc in $H_1 \setminus \{a_1\}$. Then, $(x^{[1]}, z^{[2]})$ is a feasible PCEARP tour. In the same way we can build $(x^{[2]}, z^{[3]})$ associated with arc $a_2$. If we arrange these PCEARP tours as the rows of a matrix, we obtain the block matrix shown in Figure 5.1c and, after subtracting the first row from the remaining ones, and rearranging the blocks, we obtain the full rank block matrix in Figure 5.1d. Hence, these PCEARP tours are affinely independent and, therefore, inequality $z_1 \leq x(H_1)$ induces a facet of PCEARP($G$). □

Note that, if condition (b) in Theorem 16 is not satisfied, then there is an arc $a \in H_c$ such that all the tours on $G$ traversing $a$ traverse other arcs in $H_c \setminus \{a\}$ and $x(H_c) \geq 2$ holds for all these tours. Therefore, (5.3) cannot induce a facet.

**Theorem 17.** *Connectivity inequalities* (5.2), $x(\delta^+(S)) \geq z_c - x(H_c \cap A(V \setminus S))$, *for all* $S \subset V \setminus \{1\}$ *and for all* $c \in \mathbb{H}$, *induce a facet of* PCEARP($G$) *if the following conditions are satisfied:*

*(a) graphs* $G(S)$ *and* $G(V \setminus S) \setminus H_c$ *are strongly connected,*

*(b)* $H_c \cap A_R(S) \neq \emptyset$, *and*

*(c)* $\nexists\ q \in \mathbb{H}$, $q \neq c$, *such that* $H_q \subseteq \delta(S)$

*(d) For each arc* $a \in H_c \cap A_R(V \setminus S)$, *there is a tour on* $G(V \setminus S)$ *that traverses arc* $a$ *but does not traverse any other arc in* $H_c \cap A_R(V \setminus S) \setminus \{a\}$.

*Proof.* We will do the proof for customer 1, i.e., for inequality $x(\delta^+(S)) \geq z_1 - x(H_1 \cap A(V \setminus S))$. Consider the DGRP defined on graph $G \setminus (H_1 \cap A(V \setminus S))$, where all the arcs in $A_R \setminus ((H_1 \cap A(V \setminus S)) \cup \delta(S))$ are considered as required arcs and the depot as required vertex. If conditions (a) and (b) hold, then inequality $x(\delta^+(S)) \geq 1$ is facet inducing of DGRP($G \setminus H_1 \cap A(V \setminus S)$) (see Ávila et al. (2015)) and there are $\dim(\text{DGRP}(G \setminus H_1 \cap A(V \setminus S))) = |A| - |H_1 \cap A(V \setminus S)| - |V| + 1 + 1 = p$ affinely independent DGRP tours in $G \setminus (H_1 \cap A(V \setminus S))$ satisfying $x(\delta^+(S)) = 1$. We complete these tours with a zero component for each arc in $H_1 \cap A(V \setminus S)$ to obtain $x^1, x^2, \ldots, x^p$ affinely independent DGRP tours in $G$ traversing all the arcs in $A_R \setminus ((H_1 \cap A(V \setminus S)) \cup \delta(S))$ but not traversing the arcs in $H_1 \cap A(V \setminus S)$.

If condition (b) holds, all the previous DGRP tours in $G$ traverse some arc in $H_1$ and the following are $p$ PCEARP tours satisfying $x(\delta^+(S)) = 1 = z_1 - x(H_1 \cap A(V \setminus S))$:

$$(x^1, z^{[1]}), (x^2, z^{[1]}), \ldots, (x^p, z^{[1]}),$$

where, again, $z^{[c]}$ denotes the vector with all its entries equal to zero except for $z_c = 1$.

If condition (c) holds, the DGRP tour $x^1$ (for example) can be completed with $z$-vectors as follows to obtain the following $|\mathbb{H}|-1$ PCEARP tours satisfying $x(\delta^+(S)) = 1 = z_1 - x(H_1 \cap A(V \setminus S))$:

$$(x^1, z^{[1]} + z^{[2]}), \ldots, (x^1, z^{[1]} + z^{[|\mathbb{H}|]}).$$

Furthermore, from condition (d), for each arc $a \in H_1 \cap A(V \setminus S)$, there is a tour on $G(V \setminus S)$ traversing $a$ but not traversing any other arc in $H_1 \cap A(V \setminus S)$. Let $\bar{x}^1, \ldots, \bar{x}^{|H_1 \cap A(V \setminus S)|}$ be such tours considered on $G$ (with zeros in the components corresponding to $\delta(S)$ and $G(S)$). The following $|H_1 \cap A(V \setminus S)|$ PCEARP tours satisfy $x(\delta^+(S)) = 0 = z_1 - x(H_1 \cap A(V \setminus S))$:

$$(\bar{x}^1, z^{[1]}), \ldots, (\bar{x}^{|H_1 \cap A(V \setminus S)|}, z^{[1]}).$$

Hence, we have $p + |\mathbb{H}| - 1 + |H_1 \cap A(V \setminus S)| = |A| + |\mathbb{H}| - |V| + 1$ PCEARP tours satisfying $x(\delta^+(S)) = z_1 - x(H_1 \cap A(V \setminus S))$. If we arrange them as the rows of a matrix, we obtain the block matrix shown in Figure 5.2a. After subtracting the first row from the remaining ones, and rearranging the blocks, we obtain the block matrix in Figure 5.2b, which has all its $|\mathbb{H}| + |A| - |V|$ rows linearly independent. Hence, the $|A| + |\mathbb{H}| - |V| + 1$ PCEARP tours satisfying $x(\delta^+(S)) = z_1 - x(H_1 \cap A(V \setminus S))$ are affinely independent and, therefore, inequality $x(\delta^+(S)) \geq z_1 - x(H_1 \cap A(V \setminus S))$ induces a facet of PCEARP($G$). □

### 5.2.2 Additional valid inequalities

In this section we present some inequalities that are not obtained directly from the formulation but from the properties of the PCEARP tours. These inequalities strengthen the description of the polyhedron and, therefore, contribute to improving the bounds provided by the LP relaxation of (5.1)-(5.5).

| $A \setminus H_1 \cap A(V \setminus S)$ | $H_1 \cap A(V \setminus S)$ | $z_1$ | $z_2 \ \ldots \ z_{\lvert\mathbb{H}\rvert}$ |
|---|---|---|---|
| $x^1$ <br> $\vdots$ <br> $x^p$ | 0 | 1 | 0 |
| $x^1$ <br> $\vdots$ <br> $x^1$ | 0 | 1 | I |
| $\bar{x}^1$ <br> $\vdots$ <br> $\bar{x}^{[\lvert H_1 \cap A(V \setminus S)\rvert]}$ | I | 1 | 0 |

(a)

| $z_2 \ \ldots \ z_{\lvert\mathbb{H}\rvert}$ | $H_1 \cap A(V \setminus S)$ | $z_1$ | $A \setminus H_1 \cap A(V \setminus S)$ |
|---|---|---|---|
| I | 0 | 0 | 0 |
| 0 | I | 1 | $\bar{x}^1 - x^1$ <br> $\vdots$ <br> $\bar{x}^{[\lvert H_1 \cap A(V \setminus S)\rvert]} - x^1$ |
| 0 | 0 | 1 | $x^2 - x^1$ <br> $\vdots$ <br> $x^p - x^1$ |

(b)

Figure 5.2: Matrix appearing in the proof of Theorem 17

### Parity inequalities

Parity inequalities are based on the fact that any tour crosses any cutset $\delta(S)$, $S \subseteq V \setminus \{1\}$, an even (or zero) number of times. In most of the arc routing problems defined in directed graphs, the parity inequalities are implied by the symmetry equations. However, this is not the case for the parity inequalities that we present here because they are related to some customers with arcs in the cutset and not to sets of vertices. These inequalities are the disaggregate version of those presented in Ávila et al. (2017) for the DC-CEARP.

Let $S \subseteq V \setminus \{1\}$ and a set of customers $F^{\mathbb{H}} = \{c_1, c_2, ..., c_q\}$, where $q \geq 3$ is odd and satisfying

- $H_{c_i} \cap H_{c_j} \cap \delta(S) = \emptyset$, $1 \leq i, j \leq q$, $i \neq j$, and
- $H_{c_i} \cap \delta(S) \neq \emptyset$, $1 \leq i \leq q$.

Then, the following inequality, called parity inequality,

$$x(\delta(S)) \geq \sum_{i=1}^{q} \Big( 2z_{c_i} - 1 - 2x\left(H_{c_i} \setminus \delta(S)\right) \Big) + 1. \tag{5.6}$$

is valid for the *PCEARP*:

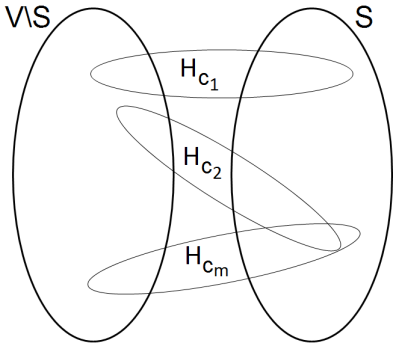



Figure 5.3: Parity inequalities (5.6)

**Theorem 18.** *Parity inequalities* (5.6) *are valid for the* PCEARP.

*Proof.* The tours servicing all the customers $c_i$ in $F^{\mathbb{H}}$, and not traversing arcs in $H_{c_i} \setminus \delta(S)$, have to traverse the cutset at least $q$ times and, since $q$ is an odd number, these tours satisfy $x(\delta(S)) \geq q + 1$, which is the inequality (5.6) with $z_{c_i} = 1$ and $x\left(H_{c_i} \setminus \delta(S)\right) = 0$.

Now consider a tour that services all customers of $F^{\mathbb{H}}$ except one, $c_j$, traversing only arcs in $\delta(S)$, and it services $c_j$ using at least one arc in $H_{c_j} \setminus \delta(S)$. This tour has to traverse the cutset at least $q - 1$ times (an even number), while the RHS of (5.6) in this case is at most $(q - 1) + (-1) + 1 = q - 1$, so it satisfies the inequality. Consider again the previous situation, but now there are two customers of $F^{\mathbb{H}}$ which are serviced by traversing arcs that are not in $\delta(S)$. This tour has to cross the cutset at least $q - 2$ times and, since $q - 2$ is an odd number, the tour satisfies $x(\delta(S)) \geq q - 2 + 1$ and, therefore, it satisfies the inequality (5.6) because, in this case, $RHS \leq (q - 2) + 2(2 - 1 - 2) + 1 = q - 3$.

The tours servicing all the customers $c_i$ in $F^{\mathbb{H}}$ but one, $c_j$, and not traversing arcs in $H_{c_i} \setminus \delta(S)$, have to traverse the cutset at least $q-1$ times and $RHS = (q-1)+(-1)+1 = q - 1$, so these tours satisfy the parity inequalities. Consider now a tour servicing all the customers in $F^{\mathbb{H}}$ but one, $c_j$, and not traversing the arcs in $H_{c_i} \setminus \delta(S)$, except for customer $c_p \in F^{\mathbb{H}}$, for which $x\left(H_{c_p} \setminus \delta(S)\right) \geq 1$. This tour crosses $\delta(S)$ at least $(q-2)+1$ times. Since the RHS of (5.6) is at most $(q-2)+(-1)+(-1)+1 = (q-3)$, the inequality is satisfied.

All the remaining cases can be proved in a similar way and are omitted here for the sake of brevity. □

**Note 1.** We do not know if the inequalities (5.6) define facets of PCEARP($G$) or under what conditions they do so. Our guess is that those conditions, if they exist, would be very restrictive. For example, if an arc $(i,j) \in \cup H_{c_i} \setminus \delta(S)$ belongs to 2 customers in $F^{\mathbb{H}}$, its variable $x_{ij}$ appears twice in (5.6) and therefore the coefficient of $x_{ij}$ is $-2-2=-4$, but it can be shown that this coefficient can be improved to -2. In general, the inequality (5.6) can be improved as follows:

$$x(\delta(S)) \geq \sum_{i=1}^{q}\Big(2z_{c_i}-1\Big) - \sum_{(i,j)\in\cup H_{c_i}\setminus\delta(S)} \alpha_{ij}x_{ij} + 1. \tag{5.7}$$

where, for each arc $(i,j) \in \cup H_{c_i} \setminus \delta(S)$ appearing in $r \geq 1$ customers, $\alpha_{ij} = r$ if $r$ is even, and $\alpha_{ij} = r+1$ if $r$ is odd.

### K-C inequalities

K-C inequalities were introduced in Corberán and Sanchis (1994) for the Undirected Rural Postman Problem. Beyond the connectivity and parity inequalities described before, the K-C inequalities try to make connectivity and parity conditions satisfied simultaneously on a partition of the vertex set that is more complex than the two shores of the cutsets $(S, V \setminus S)$ used in connectivity and parity inequalities. The name of this family of inequalities is motivated by the number of sets into which $V$ is partitioned, which is usually denoted by $K$.

Consider a partition of the set of vertices $V$ into $K$ subsets $\{M_0 \cup M_K, M_1, \ldots, M_{K-1}\}$, with $K \geq 3$, and the following set of coefficients $\alpha_{ij}$. For each $(i,j) \in A$, we define

$$\alpha_{ij} = \begin{cases} K-2, & \text{if } (i,j) \in (M_0, M_K) \\ |r-s|, & \text{if } (i,j) \in (M_r, M_s),\ \{r,s\} \neq \{0,K\} \\ 0, & \text{otherwise.} \end{cases} \tag{5.8}$$

Let $F^{\mathbb{H}} = \{c_1, c_2, ..., c_q\}$ be a set of $q$ customers, where $q \geq 2$ is even, satisfying

- $H_{c_i} \cap H_{c_j} \cap (M_0, M_K) = \emptyset$, $1 \leq i, j, \leq q$, $i \neq j$, and
- $H_{c_i} \cap (M_0, M_K) \neq \emptyset$, $1 \leq i \leq q$.

Furthermore, assume that for each $M_i$, $i = 1, \ldots, K-1$, either $1 \in M_i$ or there is a customer $n_i$, $n_i \notin F^{\mathbb{H}}$, such that the set of arcs $H_{n_i} \cap (A(M_i) \cup \delta(M_i)) \neq \emptyset$. Note that

$\delta(M_{i_1})$ and $\delta(M_{i_2})$ are not necessarily disjoint sets. Figure 5.4 depicts the structure of this inequality. We define the *K-C inequality* as:

$$\sum_{(i,j)\in A} \alpha_{ij}\ x_{ij} \geq$$
$$\geq (K-2)\sum_{i=1}^{q}\Big(2z_{c_i}-1-2x(H_{c_i}\backslash(M_0,M_K))\Big)+2\sum_{i=1}^{K-1}\Big(z_{n_i}-x(H_{n_i}\backslash(A(M_i)\cup\delta(M_i))\Big), \tag{5.9}$$

if the depot is in $M_0 \cup M_K$, and

$$\sum_{(i,j)\in A} \alpha_{ij}\ x_{ij} \geq$$
$$\geq (K-2)\sum_{i=1}^{q}\Big(2z_{c_i}-1-2x(H_{c_i}\backslash(M_0,M_K))\Big)+2\sum_{\substack{i=1\\ i\neq l.}}^{K-1}\Big(z_{n_i}-x(H_{n_i}\backslash(A(M_i)\cup\delta(M_i))\Big)+2, \tag{5.10}$$

if $1 \in M_l$ with $l \notin \{0, K\}$.

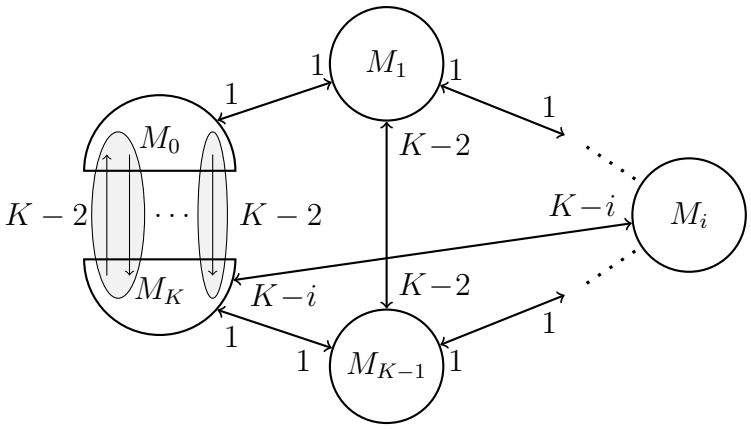



Figure 5.4: A K-C structure.

**Note 2.** If $K = 2$, then inequality (5.9) divided by two is exactly the connectivity constraint (5.2) associated with set $S = M_1$, since $x(\delta(S)) = x(\delta^+(S)) + x(\delta^-(S))$ and $x(\delta^+(S)) = x(\delta^-(S))$ hold.

The idea behind the above K-C inequalities can be understood with the following example. Consider a K-C structure with $K = 3$, $q = 2$, and the depot in $M_0$ (see Figure 5.5). Let us assume that the arcs $H_{c_1}, H_{c_2} \subset (M_0, M_K)$ and $H_{n_1} \subset A(M_1)$, $H_{n_2} \subset A(M_2)$. The K-C inequality (5.9) becomes

$$x(M_0,M_3)+x(M_0,M_1)+x(M_1,M_2)+x(M_2,M_3)+2x(M_0,M_2)+2x(M_1,M_3) \geq$$
$$\geq (2z_{c_1}-1)+(2z_{c_2}-1)+2z_{n_1}+2z_{n_2}.$$

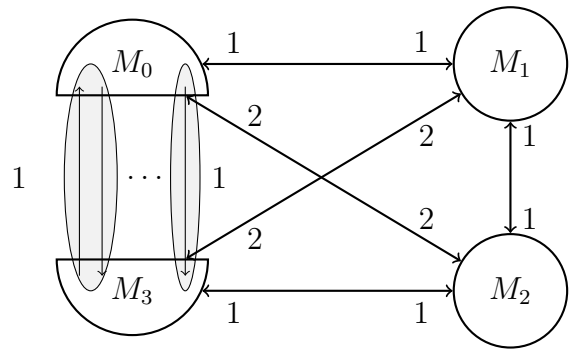



Figure 5.5: Structure of a 3-C inequality

All the PCEARP tours servicing customers $c_1, c_2, n_1$, and $n_2$ have to traverse the cutset $(M_0, M_K)$ at least twice and visit the sets $M_1$ and $M_2$. It is easy to see that all these tours have a value of the LHS of the inequality of at least 6, which is exactly the value of the RHS because all the $z$ variables take value 1. Similarly, it can be seen that all the tours servicing any subset of these customers satisfy the inequality.

**Theorem 19.** *K-C inequalities* (5.9) *and* (5.10) *are valid for the* PCEARP.

*Proof.* Let us suppose that $1 \in M_0 \cup M_K$ (the proof for the case $1 \notin M_0 \cup M_K$ is similar). We have to prove that all the PCEARP solutions $(x, z)$ satisfy inequality (5.9). We consider the following cases:

(a) Solutions $(x, z)$ servicing each customer $c_i$ from a required arc in $H_{c_i} \cap (M_0, M_K)$, $i = 1, \ldots, q$, and servicing each customer $n_i$ from a required arc in $A(M_i) \cup \delta(M_i)$, $i = 1, \ldots, K-1$. On the one hand, these tours $x$ traverse at least $q$ times the arcs in $(M_0, M_K)$, and visit at least once each node set $M_0 \cup M_K$, $M_1$, $\ldots$, $M_{K-1}$, and, hence, it can be seen that their LHS satisfy:

$$\sum_{(i,j)\in A} \alpha_{ij} x_{ij} \geq (K-2)q + 2(K-1).$$

Additionally, variables $z$ satisfy $z_{c_i} = 1$, for each $i = 1, \ldots, q$, and $z_{n_i} = 1$, for each $i = 1, \ldots, K-1$. Substituting them in the RHS of (5.9), the RHS becomes

$$(K-2)\sum_{i=1}^{q}\Big(1-2x(H_{c_i}\setminus(M_0,M_K))\Big)+2\sum_{i=1}^{K-1}\Big(1-x(H_{n_i}\setminus(A(M_i)\cup\delta(M_i))\Big) \leq (K-2)q+2(K-1).$$

Hence, these PCEARP solutions satisfy inequality (5.9).

(b) Solutions $(x, z)$ servicing each customer $c_i$ from a required arc in $H_{c_i} \cap (M_0, M_K)$, $i = 1, \ldots, q$, and servicing each customer $n_i$ from a required arc in $A(M_i) \cup \delta(M_i)$, $i = 1, \ldots, K-1$, except one of them, say $n_l$. These tours $x$ traverse at least $q$ arcs in

$(M_0, M_K)$ and visit all but one node sets $M_1, \ldots, M_{K-1}$. Note that, regarding a K-C structure (see Figure 5.4), this cannot be done at an $\alpha$-cost lower than $(K-2)q + 2(K-1) - 2$ and, hence, these tours satisfy

$$\sum_{(i,j)\in A} \alpha_{ij} x_{ij} \geq (K-2)q + 2(K-1) - 2.$$

On the other hand, variables $z$ satisfy one of the two following situations

- $z_{c_i} = 1$, for each $i = 1, \ldots, q$, and $z_{n_i} = 1$, for each $i = 1, \ldots, K-1$, except one of them, for which $z_{n_l} = 0$. Thus, if we substitute these values in the RHS of (5.9),

$$(K-2)\sum_{i=1}^{q}\Big(1 - 2x(H_{c_i} \setminus (M_0, M_K))\Big) + 2\sum_{\substack{i=1\\ i\neq l.}}^{K-1}\Big(1 - x(H_{n_i} \setminus (A(M_i) \cup \delta(M_i))\Big) -$$

$$-2x(H_{n_l} \setminus (A(M_l) \cup \delta(M_l)) \leq (K-2)q + 2(K-2).$$

- $z_{c_i} = 1$, for each $i = 1, \ldots, q$, and $z_{n_i} = 1$, for each $i = 1, \ldots, K-1$, but customer $n_l$ is serviced by traversing an arc in $H_{n_l} \setminus (A(M_l) \cup \delta(M_l))$. In this case, $x(H_{n_l} \setminus (A(M_l) \cup \delta(M_l))) \geq 1$ and the RHS of (5.9) is

$$(K-2)\sum_{i=1}^{q}\Big(1 - 2x(H_{c_i} \setminus (M_0, M_K))\Big) + 2\sum_{\substack{i=1\\ i\neq l.}}^{K-1}\Big(1 - x(H_{n_i} \setminus (A(M_i) \cup \delta(M_i))\Big) +$$

$$+2(1 - x(H_{n_l} \setminus (A(M_l) \cup \delta(M_l)))) \leq (K-2)q + 2(K-2).$$

Hence, these PCEARP solutions satisfy inequality (5.9).

(c) Solutions $(x, z)$ servicing each customer $c_i$ from a required arc in $H_{c_i} \cap (M_0, M_K)$, $i = 1, \ldots, q$, except one of them, say $c_l$, and servicing each customer $n_i$ from a required arc in $A(M_i) \cup \delta(M_i)$, $i = 1, \ldots, K-1$. These tours $x$ traverse $q-1$ (an odd number) required arcs between $M_0$ and $M_K$ and visit all the node sets $M_1, \ldots, M_{K-1}$. Again, this cannot be done at an $\alpha$-cost lower than $(K-2)(q-1)+K = (K-2)(q-2)+2(K-1)$ and, hence, these tours satisfy

$$\sum_{(i,j)\in A} \alpha_{ij} x_{ij} \geq (K-2)(q-2) + 2(K-1).$$

Moreover, variables $z$ satisfy one of the two following situations

- $z_{c_i} = 1$, for each $i = 1, \ldots, q$ except one of them, for which $z_{c_l} = 0$, and $z_{n_i} = 1$, for all $i = 1, \ldots, K-1$. Thus, the RHS of (5.9) becomes

$$(K-2)\sum_{\substack{i=1\\ i\neq l.}}^{q}\Big(1-2x(H_{c_i}\setminus(M_0,M_K))\Big)+(K-2)(-1-2x(H_{c_l}\setminus(M_0,M_K))$$
$$+2\sum_{i=1}^{K-1}\Big(1-x(H_{n_i}\setminus(A(M_i)\cup\delta(M_i))\Big)\leq(K-2)(q-1)-(K-2)+2(K-1)$$
$$\leq(K-2)(q-2)+2(K-1).$$

- $z_{c_i} = 1$, for each $i = 1, \ldots, q$, but customer $c_l$ is serviced by traversing an arc in $H_{c_l} \setminus (M_0, M_K)$, and $z_{n_i} = 1$, for all $i = 1, \ldots, K-1$. Thus, since $1 - 2x(H_{c_l} \setminus (M_0, M_K) \leq -1$, the RHS of (5.9) becomes

$$(K-2)\sum_{\substack{i=1\\ i\neq l.}}^{q}\Big(1-2x(H_{c_i}\setminus(M_0,M_K))\Big)+(K-2)(1-2x(H_{c_l}\setminus(M_0,M_K))+$$

$$+2\sum_{i=1}^{K-1}\Big(1-x(H_{n_i}\setminus(A(M_i)\cup\delta(M_i))\Big)\leq(K-2)(q-2)+2(K-1)$$

and $(x, z)$ satisfies (5.9).

(d) All the remaining cases can be proved in a similar way and are omitted here for the sake of brevity. □

## 5.3 A heuristic for the PCEARP

In this section we present an iterative algorithm we implemented to provide the exact algorithm with initial lower bounds. It combines a constructive and a local search heuristic. At each iteration, the constructive algorithm is first applied to build a set of solutions. Then, each solution (route) is possibly improved by applying the local search heuristic. The algorithm stops when a certain number of iterations without improvement is reached or a maximum computing time is exceeded. The final solution is the route associated with the best net profit among those computed. In what follows, a route $r$ is defined by a list of required arcs $A_R^r$, in the order they are traversed, and a set of serviced customers $C^r$. We will assume that the route travels from one required arc to the next (or to the depot) following the shortest path. Given $a \in A_R$, let $C_a$ be the set of customers serviced if $a$ is traversed.

At each global iteration, we first define a list of required arcs $\mathcal{A}_{ini}$ as follows: for each customer $c \in \mathbb{H}$, the arc closest to the depot among those in $H_c$ is selected and added to $\mathcal{A}_{ini}$. Then, for each arc $a_i \in \mathcal{A}_{ini}$, a route $r$ is initialized, $a_i$ is added to $A_R^r$, and all the customers in $C_{a_i}$ are included in $C^r$. Next, we check if the required arcs traversed from the depot to $a_i$ and from $a_i$ to the depot (if any) can service other unassigned customers. If so, the arcs and the corresponding customers are inserted in $A_R^r$ and $C^r$, respectively. Then, for each route $r$ we look for a subset of customers/arcs with high profits to add to $r$. In order to iteratively select the elements to be added, $\lfloor(|\mathbb{H}| - |C^r|)/4\rfloor$ customers not included in $C^r$ are randomly selected. For each selected customer $c$, we calculate the change in the objective function when any arc $a \in H_c$ is inserted in the route. The customer $\overline{c}$ and the arc $\overline{a}$ that produce the best change in the objective value are selected and $A_R^r$ and $C^r$ are updated accordingly, also taking into account the customers (if any) serviced by the arcs in the added paths when $\overline{a}$ is incorporated into $r$. For each route $r$, the selection of a new a subset of customers/arcs to insert in $r$ is repeated until a maximum number of insertions, $m\_iter\_C$, is performed without any increase in the value of the objective function. For each initial arc $a_i$, the route $r$ with the best net profit found during these $m\_iter\_C$ iterations will be kept and used in the following steps.

Each constructed solution is improved through a local search heuristic based on the destroy-and-repair. At each local search iteration, the solution $r$ is destroyed by randomly removing a subset of $m$ consecutive required arcs $A_D \subset A_R^r$ and the customers serviced by them (if they cannot be serviced by the remaining arcs in the route). Then, a repairing phase is applied. For each customer in $C^r$, we calculate the change in the objective value if the closest unassigned customer is inserted in $r$. The customer that produces the best improvement (if any) is then inserted in $C^r$. In case no new customer is inserted in $C^r$, the repairing phase stops. Otherwise, the repairing phase is reapplied with respect to the new set of customers $C^r$. The local search heuristic is repeated until $m\_iter\_LS$ iterations are performed without any improvement.

Once all the routes associated with the arcs in $\mathcal{A}_{ini}$ have been studied, if the stopping criteria have not been met, a new global iteration is performed by applying a different arc selection rule to initialize routes. For each arc $a \in \mathcal{A}_{ini}$, a customer $c$ is randomly selected from the $\max\{10, |\mathbb{H}|/50\}$ customers closest to $a$ in $\mathbb{H} \setminus C_a$. The arc $\overline{a} \in H_c$, which produces the best value of the objective function when added to the route initialized with $a$, is selected to define, together with $a$, the new initial route $r$. In this way, we have built a new list $\mathcal{A}_{ini2}$, where each component is a pair of required arcs, to initialize the routes.

The iterative algorithm (Algorithm 1) is executed for a maximum number of $iter\_max = 100 \cdot |\mathbb{H}|$ iterations and $time\_limit = 60$ seconds. The maximum number of iterations without improvement has been set to $m\_iter\_C = \min\{|\mathbb{H}|/4, 25\}$ for the constructive heuristic and to $m\_iter\_LS = \min\{|\mathbb{H}|/2, 50\}$ for the local search heuristic. The number of consecutive arcs removed while applying the destroy-and-repair mechanism has been set to $m = \min\{|A_R^r|, 5\}$.

**Input:** $G$, $\mathbb{H}$, $iter\_max$, $time\_limit$, $m\_iter\_C$, $m$, $m\_iter\_LS$
**Output:** $S_{best}$
$iter \leftarrow 0$;
$S_{best} \leftarrow \emptyset$;
Create $\mathcal{A}_{ini}$;
$\mathcal{A} \leftarrow \mathcal{A}_{ini}$;
**while** $time\_limit$ is not reached AND $iter \leq iter\_max$ **do**
  **for** each $a$ in $\mathcal{A}$ **do**
    $S_{iter} \leftarrow$ Constructive algorithm $(a, m\_iter\_C)$;
    $S_{iter} \leftarrow$ Local-Search$(m, m_iter_LS)$;
    **if** $S_{iter}$ is feasible and better than $S_{best}$ **then**
      $S_{best} \leftarrow S_{iter}$;
      $iter \leftarrow 0$;
    **else**
      $iter \leftarrow iter + 1$;
  Create $\mathcal{A}_{ini2}$;
  $\mathcal{A} \leftarrow \mathcal{A}_{ini2}$;

**Algorithm 1:** Heuristic algorithm for the PCEARP

## 5.4 A branch-and-cut algorithm for the PCEARP

We have implemented a branch-and-cut algorithm (B&C) for the PCEARP based on the formulation presented in Section 5.1 and in the separation of the connectivity (5.2), parity (5.6), and K-C inequalities (5.9) and (5.10), which are exponential in number.

### 5.4.1 Separation algorithms

The initial LP is defined by inequalities (5.3), equations (5.1), and the linear relaxation of the non-negativity and integrality conditions (5.4) and (5.5). Inequalities (5.2), (5.6), (5.9), and (5.10) are separated at each iteration of the cutting-plane algorithm and added to the LP. Let $\bar{x}, \bar{z}$ be the fractional solution obtained at an iteration of the cutting-plane algorithm. We use the following separation algorithms that have been specifically designed for this problem.

### Connectivity inequalities

Two heuristic algorithms have been used to separate connectivity inequalities. The first heuristic is based on the computation of the connected components of the graph induced by the arcs $a$ such that $\bar{x}_a \geq \varepsilon$, where $\varepsilon$ is a given parameter. For each weakly connected component with node set $S$ not including the depot, the value $\delta^+(S)$ is computed. Moreover, we compute $\bar{x}(H_c \setminus A(S))$ for each customer $c$, and select the customer $c'$ with maximum $\bar{z}_{c'} - \bar{x}(H_{c'} \setminus A(S))$. If this value is greater than $\delta^+(S)$, the inequality (5.2) associated with $S$ and $c'$ is violated. We start with $\varepsilon = 0$ and, while the algorithm fails in finding a violated inequality, we successively try $\varepsilon = 0.25$, 0.5, and 0.75.

The second heuristic is based on max-flow computations. For each customer $c$ with $\bar{z}_c \geq 0.75$, and such that $H_c$ does not contain any arc adjacent to the depot, we build a network containing the arcs for which $\bar{x}_a > 0$ (with capacity $\bar{x}_a$) and some artificial arcs (with infinity capacity) from the head vertices of the arcs in $H_c$ to an artificial sink. Then, a maximum flow from the depot to the sink is computed. This maxflow defines a minimum cut $(V \setminus S, S)$, with $1 \in V \setminus S$. Now, $\bar{z}_c - \bar{x}(H_c \setminus A(S))$ is computed for each customer and its associated inequality (5.2) is checked for violation. Since the number of violated inequalities of this type can be large and the inequalities are usually very similar, we only add those with maximum violation. If, for a given cutset, the number of such violated inequalities is greater than 30, only 30 of them are randomly selected and added.

Given a cutset $(V \setminus S, S)$ found by any of the two previous methods, if a lower bound $L$ is known, we calculate the profit $P$ associated with all the customers $c$ such that $H_c \subseteq A(V \setminus S)$. Obviously, a solution servicing only these customers would have a value of the objective function less than or equal to $P$. Therefore, if $P < L$, an optimal solution should service some other customer in $A(S) \cup \delta(S)$, so the inequality $x(\delta^+(S)) \geq 1$ has to be satisfied. Note that this inequality is stronger than the connectivity inequality (5.2) associated with a specific customer $c$, so we add the inequality in this stronger version.

### Parity inequalities

The heuristic to separate parity inequalities (5.6) has been implemented as follows. First, we compute the weakly connected components of the graph induced by the arcs satisfying $\bar{x}_a \geq 1 + \varepsilon$, if $a$ is required, and $\bar{x}_a \geq \varepsilon$, otherwise. Then, for the cutset $(V \setminus S, S)$, $1 \in V \setminus S$, associated with each connected component (each connected component is associated with node set S), we compute $\bar{x}(\delta_R(S))$ and check if this value is "close" to an odd number $r$ ($\bar{x}(\delta_R(S)) \in [r - 0.25, r + 0.25]$). If so, the

heuristic tries to select that number $r$ of "good" customers among those having arcs in the cutset. To do this, we build the set of customers $F$ with arcs in $\delta(S)$ and, for each of them, $\bar{x}(H_c \setminus \delta(S)) - \bar{z}_c$ is computed. Starting with the customer $c \in F$ with smallest $\bar{x}(H_c \setminus \delta(S)) - \bar{z}_c$ value, we iteratively select customers $c'$ from $F$ (in increasing order of the $\bar{x}(H_c \setminus \delta(S)) - \bar{z}_c$ values) such that the sets $H_{c'} \cap \delta(S)$ are disjoint with those associated with the previously selected customers, until we reach the desired number $r$ of customers. If not enough customers can be selected, we try with another component. Otherwise, the inequality (5.6) is checked for violation.

**K-C inequalities**

The main task accomplished by the heuristic for the separation of the K-C inequalities consist of finding a graph structure as the one depicted in Figure 5.4.

Consider the customers $c$ such that $\bar{x}(H_c) < z_c + \epsilon$, where $\epsilon \geq 0$ is a given parameter, and the graph induced by the arcs of these customers and the depot, if it is not already incident with one of these arcs. Let $C_i = (V_i, A_i)$ denote the connected components of this graph. For each set $V_i$, let the *exterior* vertices of $V_i$ be those vertices in $V_i$ for which there is an incident arc $a \in \delta(V_i)$ with $\bar{x}_a > 0$. Moreover, define $z_c^{net} = \bar{z}_c - \bar{x}(H_c) \leq 0$ for every customer $c$ and let $V(H_c)$ denote the set of vertices incidents with the arcs in $H_c$.

We select the customer $c_1$ with greatest value of $z_c^{net}$ such that, for some $i$, $H_{c_1} \cap A_i \neq \emptyset$ and $V(H_{c_1})$ contains at least one exterior vertex $u_1$ of $V_i$. Let $M_0 = \{u_1\}$ and $M_K = V_i \setminus \{u_1\}$. For each vertex $v \in M_K$ adjacent to a vertex in $M_0$ (initially only $u_1$), we calculate $\bar{x}((M_0 \cup \{v\}, M_K \setminus \{v\}) \cap H_{c_1})$ and add to $M_0$ (and remove from $M_K$) the vertex that maximizes this value. We repeat this process while there are vertices in $M_K$ adjacent to some vertices in $M_0$ and there is at least one exterior vertex of $V_i$ in $M_K$. Of all the studied $M_0$ (and $M_K$) sets, we keep the one that maximizes $x((M_0 \cup \{v\}, M_K \setminus \{v\}) \cap H_{c_1})$.

Given the cutset $(M_0, M_K)$, we now try to find another customer $c_2$, such that $H_{c_2} \cap (M_0, M_K) \neq \emptyset$ and $H_{c_2} \cap H_{c_1} \cap (M_0, M_K) = \emptyset$. Of all the customers $c_i$ satisfying these two conditions, we select the one that minimizes $x(H_{c_i}) - x(H_{c_i} \cap (M_0, M_K))$.

Now we shrink sets $M_0$ and $M_K$ and the remaining components $C_j$, $j \neq i$, into a single vertex each and compute a spanning tree by iteratively adding the arc of maximum weight not forming a cycle. This tree is transformed into a path linking $M_0$ and $M_K$ by (iteratively) shrinking each vertex with degree one, and not corresponding to $M_0$ and $M_K$, with its (unique) adjacent vertex. If the length of the path is at least 3, the

vertices of the path define the sets $M_0, M_1, \ldots, M_K$. All the vertices of $G$ that do not belong to a set $M_i$ yet are iteratively assigned to a set $M_i$ to which they are adjacent.

For each set $M_j$, $j = 1, \ldots, K-1$, non containing the depot, we select the customer $c_j$ that minimizes $x(H_{c_j}) - x(H_{c_j} \cap (A(M_j) \cup \delta(M_j)))$.

At this point, a K-C structure has been found, and its corresponding K-C inequality is checked for violation.

### 5.4.2 Cutting-plane algorithm

At the root node, at each iteration of the cutting-plane algorithm, the separation heuristics described above are applied in the following order:

1- The heuristic separation algorithms for connectivity inequalities. In particular, the heuristic based on flow computations is used only if the other fails to find violated inequalities.

2- The heuristic for the separation of the parity inequalities.

3- The algorithm for separating K-C inequalities. In particular this heuristic is applied only if no violated connectivity inequalities have been found by the heuristic based on the computation of connected components.

All the violated inequalities found are added to the LP relaxation.

On the remaining nodes of the branch-and-cut tree, is applied only the separation algorithm for connectivity inequalities based on the computation of connected components.

The above cutting-plane procedure is run until no new violated inequalities are found. When this happens, we branch using the Strong Branching strategy implemented in CPLEX with higher priority given to the $z_c$ variables.

In order to reduce the size of the search tree, we have used an initial lower bound obtained with the heuristic algorithm described in Section 5.3. Furthermore, we have implemented another heuristic procedure, described in the following section that provides lower bounds from the solutions of the linear relaxations.

### 5.4.3 Lower bound heuristic

Besides the heuristic algorithm described in Section 5.3, which provides an initial lower bound to the B&C, we have implemented an additional lower bound heuristic

to exploit the fractional solutions of the LPs at some nodes of the branch-and-cut tree.

Let $(\bar{x}, \bar{z})$ a fractional solution. All the customers $c$ with $\bar{z}_c \geq 0.75$ such that there is at least one arc $a \in H_c$ with $\bar{x}_a > 0.6$ are selected and their arcs with $\bar{x}_a > 0.33$ are declared as required. All the remaining arcs of $G$ are considered non-required arcs. This defines a CEARP instance that is solved with the branch-and-cut algorithm proposed in Ávila et al. (2016b) with a maximum time of 10 seconds. The best feasible solution found by this procedure is checked to see if it traverses another customer not included in the CEARP instance. If so, its variable $z_c$ is also set to 1 in the solution. We calculate the difference between the profit associated with the serviced customers and the cost of the CEARP solution, which represents a lower bound for the PCEARP optimal solution.

A the root node of the tree, this heuristic is executed every 300 iterations of the cutting-plane algorithm. After the root node has been explored, the heuristic is executed every 100 nodes up to node number 500, and every 200 nodes beyond that.

## 5.5 Computational experiments

In this section, we present the instances used to analyze the behavior of the proposed branch-and-cut algorithm, as well as the computational study performed. The algorithm has been implemented in C++ and all the tests have been run on an Intel Core i7 at 3.4 GHz with 32 GB RAM. The B&C uses CPLEX 12.10.0 with a single thread and with Concert Technology 2.9. CPLEX heuristic algorithms were turned off, and CPLEX own cuts were activated in automatic mode. The optimality gap tolerance was set to zero and best bound strategy was selected.

### 5.5.1 Instances

The performance of our algorithms has been tested on several sets of instances derived from those described in Ávila et al. (2017) for the CEARP. They are divided into two groups: *Albaida* and *Madrigueras*, whose graphs represent the street networks of two Spanish towns, and *Random*, which correspond to randomly generated instances. Note that only instances with 50 and 75 vertices of the "random" type were generated in Ávila et al. (2017). Using the same methodology, we have generated new CEARP "random" instances with up to 400 vertices. These new instances and the best solutions found for all the sets can be found at *http://www.uv.es/corberan/instancias.htm* in the classes CEARP and PCEARP. Table 5.1 shows the main characteristics of the sets of instances.

| | #Inst | $\|V\|$ | $\|A\|$ | | $\|A_R\|$ | | $\|A_{NR}\|$ | | $\|\mathbb{H}\|$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Min | Max | Min | Max | Min | Max | Min | Max |
| *Albaida* | 24 | 116 | 259 | 305 | 124 | 172 | 109 | 162 | 18 | 33 |
| *Madrigueras* | 24 | 196 | 453 | 544 | 224 | 305 | 197 | 281 | 22 | 47 |
| *Random50* | 12 | 50 | 296 | 300 | 105 | 292 | 7 | 193 | 10 | 100 |
| *Random75* | 12 | 75 | 448 | 450 | 143 | 438 | 10 | 305 | 15 | 150 |
| *Random100* | 12 | 100 | 498 | 500 | 134 | 490 | 10 | 366 | 20 | 200 |
| *Random150* | 12 | 150 | 749 | 750 | 256 | 731 | 19 | 493 | 30 | 300 |
| *Random200* | 12 | 200 | 997 | 1000 | 321 | 972 | 27 | 679 | 40 | 400 |
| *Random300* | 12 | 300 | 1498 | 1500 | 502 | 1457 | 43 | 998 | 60 | 600 |
| *Random400* | 12 | 400 | 1999 | 2000 | 675 | 1936 | 63 | 1324 | 80 | 800 |

Table 5.1: Characteristics of the sets of instances

Given that PCEARP consists of finding a tour that, servicing a subset of customers, maximizes the net profit, we have generated three different scenarios for each CEARP instance. In each scenario, the percentage of customers serviced by the optimal solution is, on average, around 60%, 80% and 90% of the total number of customers. To achieve this, we have constructed three intervals for the profits, whose means and ranges have been defined, after an extensive computational study, as follows. Given a CEARP instance $I$, let $CEARP(I)$ denote the length of the minimum cost tour servicing all customers or an upper bound for that value. $CEARP(I)$ is calculated using the exact method described in Ávila et al. (2016b). From this value, we get $\mu(I) = CEARP(I)/|\mathbb{H}|$, the average traveled distance per customer. Then, we have constructed three intervals with midpoint $\mu_\alpha(I) = \mu(I)\alpha$, where $\alpha = 0.85, 1$, and $1.15$, respectively. Their amplitude was set at 40% of $\mu(I)$. For example, for an instance $I$ and $\alpha = 0.85$, the customers are randomly generated in the interval [0.65 $\mu(I)$, 1.05 $\mu(I)$]. Table 5.2 shows the minimum and maximum profit per customer (on average) for each instance set and profit interval.

For each one of the 132 CEARP instances, we have generated three PCEARP instances for a total of 396 instances. They are grouped by set and interval profit.

### 5.5.2 Computational results

In this section we present and analyze the results obtained for the 396 PCEARP instances with the heuristic presented in Section 5.3 and the branch-and-cut algorithm. Table 5.3 summarizes the computational results obtained with the heuristic and the B&C with a time limit of 60 and 3600 seconds, respectively. For each set of instances, the table reports the number of instances in the set, the number of optima found by the heuristic, the average percentage gap between the values of the solutions provided by the heuristic and the optimal solutions (or the best lower bounds found), and the

| | [0.65$\mu$, 1.05$\mu$] | | [0.80$\mu$, 1.20$\mu$] | | [0.95$\mu$, 1.35$\mu$] | |
|---|---|---|---|---|---|---|
| | Min | Max | Min | Max | Min | Max |
| *Albaida* | 181.8 | 293.8 | 223.8 | 335.9 | 265.8 | 378.0 |
| *Madrigueras* | 167.4 | 271.0 | 206.4 | 309.7 | 245.1 | 348.6 |
| *Random50* | 93.5 | 151.3 | 115.2 | 173.1 | 136.8 | 194.5 |
| *Random75* | 80.7 | 130.5 | 99.3 | 149.3 | 118.0 | 168.0 |
| *Random100* | 70.2 | 113.9 | 86.7 | 130.1 | 102.8 | 146.3 |
| *Random150* | 61.3 | 99.2 | 75.3 | 113.2 | 89.7 | 127.6 |
| *Random200* | 49.2 | 79.5 | 60.5 | 91.2 | 72.0 | 102.5 |
| *Random300* | 38.9 | 63.2 | 48.0 | 72.3 | 57.1 | 81.3 |
| *Random400* | 36.1 | 58.8 | 44.6 | 67.3 | 53.1 | 75.8 |

Table 5.2: Average min/max profit per instance set and profit interval

average time in seconds. Moreover, for the B&C, besides the number of optima found, the table shows the average percentage gap between the upper bounds at the end of the root node (Gap0) or the final upper bounds (Gap) and the optimal solutions (or the best lower bounds found). The average number of nodes explored by the algorithm and the average computing time, in seconds, are reported in the last two columns.

| | | Heuristic | | | B&C | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | #Inst | # opt | Gap | Time | # opt | Gap0 | Gap | # nodes | Time |
| *Alb* | 72 | 59 | 0.44 | 0.9 | 72 | 0.09 | 0.00 | 1.2 | 1.0 |
| *Mad* | 72 | 39 | 3.03 | 4.0 | 72 | 1.87 | 0.00 | 10.5 | 12.3 |
| *R50* | 36 | 23 | 0.77 | 17.8 | 36 | 0.90 | 0.00 | 5.6 | 0.5 |
| *R75* | 36 | 13 | 2.38 | 29.6 | 36 | 0.32 | 0.00 | 11.1 | 1.2 |
| *R100* | 36 | 13 | 2.69 | 33.0 | 36 | 1.19 | 0.00 | 1454.4 | 51.2 |
| *R150* | 36 | 6 | 7.00 | 44.5 | 34 | 1.15 | 0.19 | 2620.4 | 269.5 |
| *R200* | 36 | 6 | 10.84 | 46.4 | 31 | 1.87 | 0.21 | 4994.8 | 763.3 |
| *R300* | 36 | 2 | 18.54 | 53.4 | 19 | 3.79 | 2.13 | 6064.1 | 2023.1 |
| *R400* | 36 | 0 | 21.67 | 60.0 | 16 | 3.09 | 2.23 | 4181.9 | 2190.8 |
| | **396** | **161** | **6.44** | **26.8** | **352** | **1.48** | **0.43** | **1759.6** | **484.2** |

Table 5.3: Heuristic and B&C results in all instances

From Table 5.3 we observe that the behavior of the heuristic is very good for small and medium instances, where it obtains a large number of optimal solutions and where it shows small gaps in short computing times. The quality of the solutions provided makes it possible for the B&C to prove their optimality or close the gap quickly. For the larger instances, the results are not so good, perhaps indicating that 60 seconds is not enough for the algorithm to find good solutions for these instances, which have up to 400 vertices, 2000 arcs, and 800 customers. On the other hand, the performance of the B&C is very good, as it is able to solve to optimality 352 instances out of 396 in less than one hour of computing time. Almost all instances with up to 200 vertices

and 1000 arcs have been solved to optimality, as well as nearly half of the largest ones (subsets *Random300* and *Random400*). The small values reported in the Gap0 column indicate that the inequalities in the formulation, together with the proposed valid inequalities, give strong linear relaxations whose optimal values are close to the values of the optimal integer solutions.

| | | Heuristic | | | B&C | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | #Inst | # opt | Gap | Time | # opt | Gap0 | Gap | # nodes | Time |
| *Alb* | 24 | 22 | 0.18 | 0.8 | 24 | 0.02 | 0.00 | 0.0 | 0.5 |
| *Mad* | 24 | 18 | 3.38 | 3.5 | 24 | 4.74 | 0.00 | 2.0 | 3.2 |
| *R50* | 12 | 8 | 1.04 | 17.7 | 12 | 1.70 | 0.00 | 4.5 | 0.5 |
| *R75* | 12 | 7 | 1.58 | 29.7 | 12 | 0.00 | 0.00 | 0.3 | 0.7 |
| *R100* | 12 | 5 | 3.57 | 32.4 | 12 | 0.92 | 0.00 | 117.3 | 7.2 |
| *R150* | 12 | 3 | 8.00 | 44.3 | 11 | 1.66 | 0.45 | 2653.3 | 313.5 |
| *R200* | 12 | 3 | 13.79 | 46.2 | 11 | 1.91 | 0.25 | 1677.2 | 346.2 |
| *R300* | 12 | 2 | 23.61 | 51.0 | 7 | 4.39 | 2.30 | 5264.6 | 1804.1 |
| *R400* | 12 | 0 | 27.58 | 60.0 | 6 | 3.65 | 2.58 | 3566.3 | 1941.9 |
| | **132** | **68** | **7.84** | **26.4** | **119** | **2.16** | **0.51** | **1208.0** | **401.9** |

Table 5.4: Heuristic and B&C results for the instances with profits in the interval $[0.65\mu, 1.05\mu]$

In order to study the influence of the profits in the behavior of the algorithms, we detail the computational results for each interval of profits in Tables 5.4-5.6. As can be seen, the heuristic behaves worse when the profit intervals are large, since in these cases (as we will see later) the average number of customers serviced increases, and with it the difficulty of the instance. Unlike the heuristic, it is not so clear that the behavior of the B&C varies with the profit intervals. The number of optima found decreases slightly while the number of explored nodes and the computing time increase. However, there is no clear pattern in the Gap and Gap0 values.

| | | Heuristic | | | B&C | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | #Inst | # opt | Gap | Time | # opt | Gap0 | Gap | # nodes | Time |
| *Alb* | 24 | 20 | 0.48 | 0.9 | 24 | 0.02 | 0.00 | 0.1 | 1.2 |
| *Mad* | 24 | 12 | 3.36 | 4.3 | 24 | 0.43 | 0.00 | 8.1 | 11.8 |
| *R50* | 12 | 7 | 1.00 | 18.0 | 12 | 0.63 | 0.00 | 3.8 | 0.5 |
| *R75* | 12 | 4 | 3.14 | 29.7 | 12 | 0.54 | 0.00 | 7.2 | 1.2 |
| *R100* | 12 | 5 | 1.98 | 33.2 | 12 | 1.69 | 0.00 | 4059.6 | 137.1 |
| *R150* | 12 | 2 | 7.41 | 44.8 | 11 | 0.92 | 0.11 | 2973.3 | 311.2 |
| *R200* | 12 | 2 | 10.75 | 46.5 | 11 | 1.81 | 0.11 | 3308.3 | 540.8 |
| *R300* | 12 | 0 | 17.30 | 54.4 | 7 | 3.30 | 1.82 | 6073.0 | 1968.7 |
| *R400* | 12 | 0 | 20.70 | 60.0 | 5 | 2.69 | 1.99 | 4333.3 | 2199.5 |
| | **132** | **52** | **6.36** | **27.0** | **118** | **1.13** | **0.37** | **1888.6** | **471.4** |

Table 5.5: Heuristic and B&C results for the instances with profits in the interval $[0.80\mu, 1.20\mu]$

| | | Heuristic | | | B&C | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | #Inst | # opt | Gap | Time | # opt | Gap0 | Gap | # nodes | Time |
| *Alb* | 24 | 17 | 0.65 | 0.9 | 24 | 0.24 | 0.00 | 3.4 | 1.3 |
| *Mad* | 24 | 9 | 2.36 | 4.3 | 24 | 0.44 | 0.00 | 21.5 | 21.9 |
| *R50* | 12 | 8 | 0.27 | 17.7 | 12 | 0.37 | 0.00 | 8.5 | 0.5 |
| *R75* | 12 | 2 | 2.41 | 29.3 | 12 | 0.43 | 0.00 | 25.8 | 1.8 |
| *R100* | 12 | 3 | 2.53 | 33.4 | 12 | 0.96 | 0.00 | 186.3 | 9.2 |
| *R150* | 12 | 1 | 5.59 | 44.3 | 12 | 0.87 | 0.00 | 2234.6 | 184.0 |
| *R200* | 12 | 1 | 7.98 | 46.5 | 9 | 1.89 | 0.29 | 9998.8 | 1402.8 |
| *R300* | 12 | 0 | 14.73 | 54.7 | 5 | 3.68 | 2.28 | 6854.8 | 2296.6 |
| *R400* | 12 | 0 | 16.73 | 60.0 | 5 | 2.94 | 2.13 | 4646.3 | 2430.9 |
| | **132** | **41** | **5.11** | **26.9** | **115** | **1.14** | **0.43** | **2182.3** | **579.3** |

Table 5.6: Heuristic and B&C results for the instances with profits in the interval $[0.95\mu, 1.35\mu]$

To study in more depth if the difficulty of the problem varies with the percentage of customers serviced by the solution (due to the different profit intervals), we have summarized in Table 5.7 the results obtained by the branch-and-cut algorithm on the largest instances, grouped by number of customers and profit interval. We want to point out that, although the table reports some of the same measures in the previous tables, the gap values have been calculated here only for the instances not solved to optimality in the corresponding set, while the computing times are calculated only for those optimally solved (in less than 3600 seconds). The table also shows the average percentage of customers serviced by the optimal solution or by the best solution found ($\%c$).

From Table 5.7 we observe that the computing time needed to solve the instances to optimality grows with the profit interval, which is a clear indication of an increase in the difficulty of the instances at the increase of the percentage of customers serviced. The number of optima found also decreases as this percentage increases, although it is only a slight decrease. The behavior of the gaps is not uniform. While we could expect the largest gaps for the unsolved instances with the highest number of customers serviced, their values do not seem to follow this trend. On the other hand, we can see that the total number of customers ($|\mathbb{H}|$) significantly influences the difficulty of the instance.

## 5.6 Conclusions

In this chapter, we have addressed a new arc routing problem, the Profitable Close-Enough Arc Routing Problem (PCEARP). In the PCEARP, customers are not necessarily nodes of a network, but can be serviced by traversing an edge that is close

| | | [0.65$\mu$, 1.05$\mu$] | | | | [0.80$\mu$, 1.20$\mu$] | | | | [0.95$\mu$, 1.35$\mu$] | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\lvert\mathbb{H}\rvert$ | opt/inst | Gap | Time | %$c$ | opt/inst | Gap | Time | %$c$ | opt/inst | Gap | Time | %$c$ |
| *R150* | 30 | 3/3 | - | 1.7 | 51.3 | 3/3 | - | 2.1 | 77.7 | 3/3 | - | 2.5 | 81.0 |
| | 75 | 3/3 | - | 4.4 | 65.0 | 3/3 | - | 5.8 | 73.3 | 3/3 | - | 9.5 | 81.7 |
| | 150 | 3/3 | - | 13.7 | 70.7 | 3/3 | - | 22.9 | 78.7 | 3/3 | - | 44.4 | 84.7 |
| | 300 | 2/3 | 5.45 | 51.1 | 80.7 | 2/3 | 1.33 | 20.95 | 87.0 | 3/3 | - | 679.5 | 87.0 |
| *R200* | 40 | 3/3 | - | 2.9 | 36.7 | 3/3 | - | 9.4 | 66.7 | 3/3 | - | 13.9 | 83.3 |
| | 100 | 3/3 | - | 32.5 | 65.3 | 3/3 | - | 93.4 | 77.3 | 3/3 | - | 155.3 | 79.0 |
| | 200 | 3/3 | - | 78.3 | 70.7 | 3/3 | - | 309.6 | 81.3 | 3/3 | - | 1842.0 | 84.0 |
| | 400 | 2/3 | 2.96 | 106.7 | 80.7 | 2/3 | 1.27 | 826.4 | 83.3 | 0/3 | 1.15 | - | 88.7 |
| *R300* | 60 | 3/3 | - | 14.6 | 40.7 | 3/3 | - | 59.5 | 70.7 | 3/3 | - | 162.2 | 76.7 |
| | 150 | 3/3 | - | 230.7 | 57.3 | 3/3 | - | 628.2 | 74.7 | 2/3 | 1.77 | 936.0 | 82.0 |
| | 300 | 1/3 | 8.55 | 2912.9 | 69.3 | 0/3 | 5.58 | - | 81.3 | 0/3 | 3.48 | - | 83.0 |
| | 600 | 0/3 | 3.51 | - | 78.0 | 1/3 | 2.54 | 3555.6 | 85.7 | 0/3 | 5.05 | - | 87.0 |
| *R400* | 80 | 3/3 | - | 103.1 | 56.0 | 3/3 | - | 189.6 | 69.7 | 3/3 | - | 487.0 | 81.3 |
| | 200 | 2/3 | 3.19 | 156.7 | 61.3 | 2/3 | 0.98 | 303.6 | 76.7 | 2/3 | 1.77 | 1244.8 | 85.0 |
| | 400 | 1/3 | 3.51 | 1060.2 | 75.0 | 0/3 | 2.88 | - | 83.3 | 0/3 | 2.87 | - | 85.7 |
| | 800 | 0/3 | 6.91 | - | 28.3 | 0/3 | 4.74 | - | 58.3 | 0/3 | 5.06 | - | 89.3 |
| | | **35/36** | **-** | **-** | **61.7** | **34/36** | **-** | **-** | **76.6** | **31/36** | **-** | **-** | **83.7** |

Table 5.7: B&C results, grouped by interval profits, for the largest sets of instances

enough. A profit is associated with each customer and it is collected only once when the customer is serviced. The objective is to find a tour maximizing the net profit, that is, the difference between the total profit collected and the distance traveled.

In this work, we have presented a formulation for the PCEARP, as well as some valid inequalities, and studied the polyhedron associated with its solutions. To solve the PCEARP, we have proposed a heuristic algorithm that provides good quality lower bounds in short computing times. A branch-and-cut using the knowledge obtained from the polyhedral study and the lower bounds calculated by the heuristic algorithm have also been presented. This exact algorithm is able to optimally solve large instances with up to 600 customers, 300 vertices, and 1500 arcs in one hour of computing time.

# Chapter 6

# A Matheuristic for the Distance-Constrained Close-Enough Arc Routing Problem

## 6.1 Introduction

In this chapter we study the Distance-Constrained Close-Enough Arc Routing Problem (DC-CEARP) introduced in Ávila et al. (2017), a generalization of the CEARP in which a fleet of vehicles, or a vehicle multiple times, performs the service of the customers. This problem consists of finding a set of $K$ routes leaving from and entering at the depot and serving all the customers, such that the length (in distance or time) of each route does not exceed a certain value $D_{max}$. The objective is to minimize the total length traversed. Ávila et al. noted that the DC-CEARP is NP-*hard*, since it generalizes the CEARP. For this problem, several formulations and exact algorithms were proposed and compared on a set of instances with up to five vehicles, 196 vertices, 450 arcs, and 150 customers.

For this problem we propose a multi-start matheuristic that incorporates an effective branch-and-cut method for the CEARP in order to optimize the routes obtained. In Section 6.2 we define formally the DC-CEARP and introduce the notation used. Moreover, the most promising formulation of the problem, according to Ávila et al. (2017), is given. The matheuristic is presented in Section 6.3 and the computational experiments are described in Section 6.4. Finally, some conclusions are given in Section 6.5.

## 6.2 Problem definition and notation

The DC-CEARP is defined on a strongly connected directed graph $G = (V, A)$, where $V$ denotes the set of vertices and $A$ the set of arcs. Vertex 1 is the depot, and, for each arc $(i, j) \in A$, there is a distance or length $d_{ij}$ associated with its traversal. Let us call $\mathbb{H} = \{1, \ldots, L\}$ the set of customers and $H_c \subseteq A$ the associated set of arcs from which each customer $c$ can be serviced. We assume there is a homogeneous fleet of $K$ vehicles based at the depot. The duration of the routes must not exceed a maximum travel distance denoted by $D_{max}$. The objective of the DC-CEARP is to find a set of $K$ routes, starting and ending at the depot, with minimum total cost and such that at least one arc form each $H_c$, $c \in \mathbb{H}$, is traversed by the routes.

In what follows, $\mathbb{K} = \{1, \ldots, K\}$ will represent the set of vehicles and $A_R = H_1 \cup H_2 \cup \ldots \cup H_L$ the set of required arcs. The arcs in the set $A_{NR} = A \setminus A_R$ are called non-required arcs. Given a set $S \subset V$, we define $\delta^+(S) = \{(i, j) \in A : i \in S,\ j \in V \setminus S\}$, $\delta^-(S) = \{(i, j) \in A : i \in V \setminus S,\ j \in S\}$ and $A(S) = \{(i, j) \in A : i, j \in S\}$. Finally, given a set $F$ of arcs, $x^k(F) = \sum_{(i,j)\in F} x^k_{ij}$.

In the paper by Ávila et al. (2017) four different formulations for the DC-CEARP were presented. The formulation that provided the best computational results was called $F_{xz}$ and is presented in what follows. Let us define the variables:

$x^k_{ij}$ = number of times that the vehicle $k$ traverses arc $(i, j) \in A$,

$$z^k_c = \begin{cases} 1, & \text{if the customer } c \text{ is serviced by vehicle } k \\ 0, & \text{otherwise.} \end{cases}$$

The problem can be formulated as

s.t.:

$$\text{Minimize} \sum_{k\in\mathbb{K}} \sum_{(i,j)\in A} d_{ij}\, x^k_{ij}$$

$$x^k(\delta^+(i)) = x^k(\delta^-(i)) \qquad \forall i \in V,\ \forall k \in \mathbb{K} \tag{6.1}$$

$$x^k(\delta^+(S)) \geq z^k_c - x^k(H_c \cap A(V \setminus S)) \qquad \forall S \subset V \setminus \{1\},\ \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{6.2}$$

$$\sum_{k\in\mathbb{K}} z^k_c = 1 \qquad \forall c \in \mathbb{H} \tag{6.3}$$

$$\sum_{(i,j)\in H_c} x^k_{ij} \geq z^k_c \qquad \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{6.4}$$

$$\sum_{(i,j)\in A} d_{ij}\, x^k_{ij} \leq D_{max} \qquad \forall k \in \mathbb{K} \tag{6.5}$$

$$x^k_{ij} \geq 0 \text{ and integer} \qquad \forall (i, j) \in A,\ \forall k \in \mathbb{K} \tag{6.6}$$

$$z^k_c \in \{0, 1\} \qquad \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{6.7}$$

Inequalities (6.1) are the symmetry conditions on the vertices. Constraints (6.2) ensure the connectivity of the routes. Note that, if vehicle $k$ does not service customer $c$, $z_c^k = 0$, or it services customer $c$ by traversing an arc in $H_c \cap A(V \setminus S)$, then the inequality is trivially satisfied. On the other hand, if vehicle $k$ services customer $c$ by traversing an arc not in $H_c \cap A(V \setminus S)$, it has to traverse the cutset $\delta(S)$. The mandatory service of the customers is established in inequalities (6.3), and inequalities (6.4) link the two sets of variables. Finally, constraints (6.5) limit the duration of the routes and (6.6) and (6.7) are the non-negativity and integrality constraints.

Using this formulation, Ávila et al. (2017) implemented a branch-and-cut algorithm capable of solving instances with up to five vehicles, 196 vertices, 450 arcs, and 150 customers. Nevertheless, the algorithm could not find the optimal solution in 30 out of the 251 instances and in seven of these 30 was not even able to produce a feasible solution after one hour of computing time. This, and the need of solving larger instances, as well as to provide good upper bounds that can help the exact algorithm, motivated us to develop a heuristic algorithm for the DC-CEARP.

## 6.3 Matheuristic

The proposed algorithm is a multi-start matheuristic. In each iteration of the algorithm, the routes are first generated using a heuristic construction algorithm and they are then improved by two local search procedures. Finally, each route is optimized by means of an exact procedure. The algorithm stops when a certain number of iterations with no improvement is reached or a maximum computation time is exceeded.

Before applying this algorithm, we first remove those customers $c_1$ such that there is another customer $c_2$ for which $H_{c_2} \subseteq H_{c_1}$.

### 6.3.1 Constructive algorithm

In this section we describe how to construct feasible solutions for the DC-CEARP by iteratively assigning each customer to a route. Since the total duration of the routes could be improved in later phases of the algorithm, we allow the duration of the constructed routes to exceed the time limit by a certain amount given by a parameter *Marg* varying between 0 and 0.2. Let $D_L = (1+Marg) \times D_{max}$ be the maximum duration allowed for the routes in this phase of the algorithm.

A route for vehicle $i$ will be represented as the sequence $R_i$ of required arcs traversed by vehicle $i$. We assume that, in order to travel from the end of a required arc to the beginning of the following one, the vehicle uses the shortest path. Associated with

each route $i$, we have a vector $C_i$ containing the customers which are serviced from the required arcs in $R_i$.

The constructive algorithm consists of two phases: the first one initializes the routes, and the second one completes them by iteratively allocating the customers not yet assigned to any route. We have implemented six variants of the constructive method, resulting from the combination of three different initialization procedures with two completion methods that are described in what follows.

**Initialization procedures**

We define the service time of a customer $c$, and denote it $st_c$, as the shortest time needed for traveling from the depot to an arc in $H_c$, traversing it, and returning to the depot. Based on this service time, we have considered different ways of introducing the first arc in each route.

- *Random Initialization*
  To initialize a route $i$, we randomly select a customer $c$ among those that have not been assigned to any route yet. Among all the arcs in $H_c$, we select the arc $a_c$ associated with the minimum service time $st_c$. Likewise, $a_c$ is introduced in $R_i$, and both $c$ and the rest of unassigned customers that can be serviced by $a_c$ are included in $C_i$.

- *Random Selection among the Best Applicants*
  We define $BA$ (Best Applicants) as the set formed by the $min\{10, \frac{|\mathbb{H}|}{2}\}$ unassigned customers with the highest $st_c$ values.

  To initialize the route $i$, we randomly select a customer $c \in BA$ and its corresponding arc $a_c \in H_c$ with minimum service time $st_c$ is inserted in $R_i$. Customer $c$, together with all the other unassigned customers that can be serviced by arc $a_c$, is added to $C_i$.

- *Weighted Selection among the Best Applicants*
  A customer $c$ is randomly chosen from the set $BA$ defined above using probabilities
  $$p_c = \frac{st_c}{\sum_{i \in BA} st_i}.$$
  Customer $c \in BA$ is introduced in $C_i$ and the arc $a_c \in H_c$ associated with $st_c$ is included in $R_i$. Again, all the unassigned customers that can be serviced by $a_c$ are included in $C_i$.

Note that inserting an arc in an empty route implies traveling from the depot to this arc and returning back using shortest paths. These shortest paths are studied to

check if they traverse any arcs from which unassigned customers could be serviced. If this is the case, these customers are added to the corresponding route and marked as assigned, and the arcs from which they are serviced are marked as required and added to the route in the corresponding position.

### Route completion procedures

Once the routes have been initialized using one of the variants explained above, a deterministic completion procedure is used to introduce the remaining customers in the routes by iteratively adding arcs to the partial routes. Note that an arc that is inserted in a route $i$ may also service other unassigned customers. These customers are also added to $C_i$. Likewise, inserting an arc in a route may result in changing the shortest paths traversed between required arcs. If these new shortest paths traverse any required arc that can service unassigned customers, these customers are also added to $C_i$, and the corresponding required arc is included in $R_i$.

Two different completion methods are used: one completing all the routes simultaneously and another one working in a sequential way, completing each route before moving to the next one.

- *Parallel Completion*
  Once all the routes have been initialized, we start by allocating the unassigned customers $c$ with $|H_c| = 1$, since these are the customers with less flexibility to be inserted in the routes. For each of these customers we check all the possible positions and routes in which its corresponding arc can be inserted, and choose the one that produces a minimum increase of the route duration.

  For each arc $a$ of the remaining unassigned customers, we calculate how much the duration of the route increases for all the possible positions and all the possible routes in which we can insert $a$. The minimum of these values is called the insertion cost of $a$. Note that an insertion in a route is considered possible if the resulting route time does not exceed $D_L$. For each unassigned customer $c$, let $a_c^*$ be the arc in $H_c$ with minimum insertion cost. Then, we choose the customer $c^*$ with the maximum insertion cost $a_{c^*}^*$. Arc $a_{c^*}^*$ is inserted in the best possible position and route, say $R_i$, and $c^*$ is added to $C_i$.

- *Sequential Completion*
  In this case, after a route $i$ is initialized, it is completed until no more customers can be allocated to it without exceeding $D_L$.

  For each unassigned customer, we obtain the arc and the position in which it should be introduced so that the increase of the duration of the route is

minimum. This process is repeated until no customer can be inserted in the route without exceeding $D_L$, i.e, the current route has been completed. If there are some customers that are not allocated yet to any route and there is an empty route, a new route is initialized.

### 6.3.2 Local search

The solutions obtained in the construction phase are improved using two local search procedures. The methods *2-Exchange*, based on the exchange of two arcs from different routes, and *destroy and repair*, where some arcs and their corresponding customers are removed from the solution and then reintroduced, are described.

#### 2-Exchange

The *2-Exchange* procedure consists of swapping two arcs of different routes. Note that when two arcs of different routes are exchanged, the customers associated with each arc must be exchanged too. As it is explained below, this exchange is applied following the *first improvement* strategy.

The routes are sorted randomly and the first two routes $i$ and $j$ are selected. Then, an arc $a_i^*$ from $R_i$ and an arc $a_j^*$ from $R_j$ are deleted and the customers serviced from $a_i^*$ and $a_j^*$ are removed from $C_i$ and $C_j$, respectively. Arc $a_j^*$ is placed in the best possible position of the route $i$, and arc $a_i^*$ in the best possible position of the route $j$. The customers serviced from the arc $a_j^*$ are included now in $C_i$, and those serviced from $a_i^*$ are included in $C_j$. If the obtained solution improves the current one and $D_L$ is not exceeded, the movement is accepted and another pair of routes is selected. Otherwise, the movement is discarded and another pair of arcs of $R_i$ and $R_j$ is chosen until an improving movement is found or all the possible exchanges for this pair of routes have been tried.

#### Destroy and repair

This method consists of two phases: in the first one, some arcs are removed from the current solution, while in the second one of the customers that cannot be serviced due to the removed arcs are reallocated.

In the destruction phase of the algorithm, $m$ arcs are removed from the solution but always keeping at least one arc in any route. Specifically, an arc is randomly selected and, if it is not the only arc in its route, it is removed. Let $a_i$ be an arc removed from $R_i$ and $c$ a customer serviced from this arc. If there is another arc in $R_i$ from which customer $c$ can be serviced, $c$ does not need to be reallocated. If this is not the case

but there is an arc in another route $R_j$ from which $c$ can be serviced, $c$ is reassigned to route $R_j$ without modifying the routes. Otherwise, customer $c$ will be marked as unassigned in order to be reallocated by the repair phase.

Once the destruction phase is finished, all the unassigned customers must be reallocated. This is done using the *parallel completion* procedure described in Section 6.3.1. If the resulting solution does not improve the original one, all the changes made in this phase are discarded. This procedure is repeated for each value of $m \in \{1, 2, 5\}$ until $min\{50, \frac{|\mathbb{H}|}{2}\}$ iterations without an improvement are performed.

### 6.3.3 Route optimization

Finally, we optimize each single route by solving a single-vehicle CEARP problem using the exact branch-and-cut algorithm developed in Ávila et al. (2016b).

Given a route $i$, we define a CEARP instance on graph $G$ with the set of customers $C_i$. Each customer $c \in C_i$ has the same associated set of required arcs $H_c$ as in the original DC-CEARP instance. This CEARP instance is then solved optimally. If the duration of the optimal route is greater than $D_{max}$, the route is not feasible and the current solution is discarded.

If the obtained route is feasible, it may now traverse some required arcs that service customers which are currently assigned to other routes. If we can now service a new customer $c$ that belongs to another route $j$ that has not been optimized yet, we add $c$ to $C_i$ and remove it from $C_j$. Since the order of selecting the routes to be optimized can have an effect in the optimization of the remaining routes, the selection is made at random.

### 6.3.4 Overall algorithm

The multi-start matheuristic consists basically of generating solutions using the different constructive algorithms presented above and applying the local-search procedures and the exact optimization to the individual routes. This procedure is repeated until a certain stopping criterium is met. Specifically, the matheuristic (Algorithm 2) is initialized by creating a solution with one of the six constructive algorithms proposed. Note that, since this solution is built using a time or distance limit for the routes $D_L = (1+Marg) \times D_{max}$, if $Marg > 0$, the duration of some routes may exceed $D_{max}$ and thus the solution may not be feasible. The obtained solution is improved by means of the local-search algorithm and then each route is optimized using the procedure described in Section 6.3.3. This construction and improvement phase is repeated for all the different constructive algorithms. If the best feasible solution

(if any) obtained in this way is better than the current best solution, it is stored as the new best solution. This procedure is repeated until a maximum computing time $time\_limit$ is exceeded or a certain number of iterations $iter\_max$ without updating the best known solution is reached.

The overall algorithm (Algorithm 3) applies the multi-start procedure with values $Marg = 0, 0.05, 0.1$ and a maximum number of iterations without improvement of $iter\_max1$. If no feasible solution has been found, the multi-start procedure is used again with maximum number of iterations $iter\_max2$ and increasing the value of $Marg$ by 0.02 iteratively. The procedure stops when a feasible solution is found, $Marg > 0.2$, or the computing time $time\_limit$ is exceeded, whatever happens first.

**Input:** $G$, $\mathbb{H}$, $D_{max}$, $Marg$, $iter\_max$, $time\_limit$
**Output:** $S_{best}$
$D_L \leftarrow (1+Marg) \times D_{max}$;
$iter \leftarrow 0$;
$S_{best} \leftarrow \emptyset$;
**while** $time\_limit$ is not reached AND $iter \leq iter\_max$ **do**
  $S_{iter} \leftarrow \emptyset$;
  **for** each Constructive algorithm **do**
    $S_c \leftarrow$ Constructive algorithm$(D_L)$;
    $S_i \leftarrow$ Local-Search$(S_c, D_L)$;
    $S_o \leftarrow$ Routes optimization$(S_i, D_{max})$;
    **if** $S_o$ is feasible and better than $S_{iter}$ **then**
      $S_{iter} \leftarrow S_o$;
  **if** $S_{iter}$ is feasible and better than $S_{best}$ **then**
    $S_{best} \leftarrow S_{iter}$;
    $iter \leftarrow 0$;
  **else**
    $iter \leftarrow iter + 1$;

**Algorithm 2:** Multi-start

**Input:** $G$, $\mathbb{H}$, $D_{max}$, $iter\_max1$, $iter\_max2$, $time\_limit$
**Output:** $S_{best}$
$Marg \leftarrow 0$;
$S_{best} \leftarrow \emptyset$;
**while** $time\_limit$ is not reached AND $Marg \leq 0.1$ **do**
  $S_{best} \leftarrow$ Multi-start($G$, $\mathbb{H}$, $D_{max}$, $Marg$, $iter\_max1$, $time\_limit$);
  $Marg \leftarrow Marg + 0.05$;
**if** $S_{best} = \emptyset$ **then**
  **while** $S_{best} = \emptyset$ AND $time\_limit$ is not reached AND $Marg \leq 0.2$ **do**
    $S_{best} \leftarrow$ Multi-start($G$, $\mathbb{H}$, $D_{max}$, $Marg$, $iter\_max2$, $time\_limit$);
    $Marg \leftarrow Marg + 0.02$;

**Algorithm 3:** Overall

## 6.4 Computational experiments

In this section we study the performance of the proposed algorithm. The instances tried and the computational results obtained are described in the following sections. The procedures have been coded in C++ and all the computational experiments have been executed on a single thread of an Intel Core i7 at 3.4GHz with 32GB RAM running Windows 10 Enterprise 64 bits. The branch-and-cut algorithm for the single-vehicle CEARP used for optimizing the routes (see Section 6.3.3), developed in Ávila et al. (2016b), was also implemented in C++ using CPLEX 12.4 MIP Solver with Concert Technology 2.9. Although this exact procedure is able to solve large-sized CEARP instances, it can be quite time consuming. Therefore, we have limited its execution time to 10 seconds.

### 6.4.1 Instances

The algorithm has been tested on the four sets of instances proposed in Ávila et al. (2017). The instances of the two first sets, called *Random50* and *Random75*, were generated randomly in a 1000 x 1000 square, with $|V| \in \{50, 75\}$. The other two sets of instances, called *Albaida* and *Madrigueras* are based on the street networks of these two Spanish towns. There is a total of 251 instances whose characteristics are shown in Table 6.1. All the data, including the values of $D_{max}$ and the number of vehicles, as well as the detailed results described in Section 6.4.2, can be found in *http://www.uv.es/corberan/instancias.htm*.

| | #Inst | $\|V\|$ | $\|A\|$ | | $\|A_R\|$ | | $\|A_{NR}\|$ | | $\|\mathbb{H}\|$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Min | Max | Min | Max | Min | Max | Min | Max |
| *Albaida* | 24 | 116 | 259 | 305 | 124 | 172 | 109 | 162 | 18 | 33 |
| *Madrigueras* | 24 | 196 | 453 | 544 | 224 | 305 | 197 | 281 | 22 | 47 |
| *Random50* | 12 | 50 | 296 | 300 | 105 | 292 | 7 | 193 | 10 | 100 |
| *Random75* | 12 | 75 | 448 | 450 | 143 | 438 | 10 | 305 | 15 | 150 |

Table 6.1: Characteristics of the instances

### 6.4.2 Computational results

The results obtained with two versions of the matheuristic are compared to those obtained with the branch-and-cut in Ávila et al. (2017). Table 6.2 summarizes the results obtained with a version of the algorithm with $iter\_max1 = 5$, $iter\_max2 = 20$, and $time\_limit = 100$ (seconds), which will be called *Matheuristic 1* (MH1). The goal of *Matheuristic 1* is to obtain good solutions in low computing times. Table 6.2 shows the results obtained only for the instances for which the branch and cut

obtained an optimal solution. Columns 1 and 2 contain the name of the instance set and the number of vehicles. Column 3 reports the number of instances with known optimal solution. Column 4 shows the number of instances for which *Matheuristic 1* reached the known optimal solution, while Column 5 shows the number of instances for which the algorithm found feasible but not optimal solutions. For the instances reported in Column 5, we have computed the gap between the cost of the solution provided by *Matheuristic 1* and the optimal value. The average gap values are shown in Column 6. The number of instances for which the algorithm was not even capable of finding a feasible solution is given in Column 7. The last two columns report the average time in seconds taken by *Matheuristic 1* and the branch-and-cut, respectively, for all the instances. Note that the times reported for *Matheuristic 1* are lower than $time\_limit = 100$. This is due to the values used for parameters $iter\_max1$ and $iter\_max2$ limiting the running time of the algorithm.

Globally, the algorithm *Matheuristic 1* has been capable of reaching the optimal solution on 115 out of 221 instances, in 22.9 seconds on average. The average gap from the optimal value in the 101 instances for which a feasible solution was found is 3.93%. However, there are 5 instances for which *Matheuristic 1* has not been able to find a feasible solution with the given number of vehicles. This may be explained by the fact that the values of $D_{max}$ limiting the length of the routes are very tight. As expected, the *Random75* instances are more difficult to solve to optimality than the *Random50* ones because of their larger sizes. Something similar happens with the *Madrigueras* instances with respect to the *Albaida* ones, although, perhaps due to the structure of their underlying graphs, the number of instances for which no feasible solution has been found is greater in the *Albaida* set than in the *Madrigueras* set.

In order to obtain better solutions, we have tried another version of the algorithm, called *Matheuristic 2*, with the following values of the parameters: $iter\_max1 = iter\_max2 = 200$ and $time\_limit = 600$ (seconds). The obtained results are shown in Table 6.3. The reader can observe the different behavior of *Matheuristic 2* against *Matheuristic 1*. Now, the number of optimal solutions found is 161 out of 221 versus 115 of the faster version. Also, the average gap for the instances that were not solved optimally is 2.41% against 3.93% of *Matheuristic 1*. Finally, only in 2 out of the 221 instances was *Matheuristic 2* not capable of finding a feasible solution with the specified number of vehicles. Of course, all these better results have been obtained at the expense of a greater computational effort (151 seconds on average versus 22 seconds of *Matheuristic 1*).

Table 6.4 shows the computational results obtained on the instances of the *Random50*, *Random75*, and *Madrigueras* sets with unknown (or unproven) optimal solutions. In

| | #Veh | #Inst | #Opt | #No Opt | Gap(%) | #No Sol | Time (s) MH1 | Time (s) B&C |
|---|---|---|---|---|---|---|---|---|
| | 2 | 12 | 8 | 4 | 3.76 | 0 | 1.9 | 45.9 |
| | 3 | 11 | 6 | 5 | 4.67 | 0 | 3.1 | 83.0 |
| Random50 | 4 | 9 | 4 | 5 | 4.76 | 0 | 4.3 | 179.3 |
| | 5 | 2 | 0 | 2 | 4.01 | 0 | 12.3 | 456.4 |
| | | **34** | **18** | **16** | **4.39** | **0** | **3.6** | **117.4** |
| | 2 | 12 | 6 | 6 | 4.30 | 0 | 2.6 | 388.5 |
| | 3 | 12 | 5 | 7 | 4.40 | 0 | 6.6 | 603.1 |
| Random75 | 4 | 10 | 3 | 7 | 3.94 | 0 | 13.3 | 771.1 |
| | 5 | 4 | 2 | 2 | 3.82 | 0 | 5.6 | 1301.1 |
| | | **38** | **16** | **22** | **4.17** | **0** | **7.0** | **654.3** |
| | 2 | 24 | 21 | 3 | 0.76 | 0 | 14.9 | 34 |
| | 3 | 24 | 18 | 6 | 5.90 | 0 | 21.0 | 89.9 |
| Albaida | 4 | 21 | 9 | 12 | 3.17 | 0 | 22.9 | 338.7 |
| | 5 | 17 | 10 | 3 | 3.43 | 4 | 40.1 | 316.1 |
| | | **86** | **59** | **25** | **3.59** | **4** | **23.5** | **182.1** |
| | 2 | 24 | 11 | 13 | 2.82 | 0 | 39.0 | 224.1 |
| | 3 | 21 | 6 | 15 | 3.96 | 0 | 43.1 | 894.1 |
| Madrigueras | 4 | 13 | 4 | 9 | 5.20 | 0 | 39.8 | 1625.2 |
| | 5 | 5 | 2 | 2 | 3.08 | 1 | 58.6 | 2447.2 |
| | | **63** | **23** | **39** | **3.82** | **1** | **42.1** | **912.9** |
| **Total** | | **221** | **115** | **101** | **3.93** | **5** | **22.9** | **462.9** |

Table 6.2: Results of *Matheuristic 1* for the instances with known optimal solution

| | #Veh | #Inst | #Opt | #No Opt | Gap(%) | #No Sol | Time (s) MH2 | Time (s) B&C |
|---|---|---|---|---|---|---|---|---|
| | 2 | 12 | 9 | 3 | 4.48 | 0 | 13.5 | 45.9 |
| | 3 | 11 | 9 | 2 | 0.42 | 0 | 26.9 | 83.0 |
| Random50 | 4 | 9 | 5 | 4 | 2.08 | 0 | 21.1 | 179.3 |
| | 5 | 2 | 1 | 1 | 0.57 | 0 | 35.2 | 456.4 |
| | | **34** | **24** | **10** | **2.32** | **0** | **21.1** | **117.4** |
| | 2 | 12 | 8 | 4 | 2.67 | 0 | 22.1 | 388.5 |
| | 3 | 12 | 6 | 6 | 3.20 | 0 | 26.2 | 603.1 |
| Random75 | 4 | 10 | 5 | 5 | 3.41 | 0 | 36.1 | 771.1 |
| | 5 | 4 | 2 | 2 | 2.57 | 0 | 34.8 | 1301.1 |
| | | **38** | **21** | **17** | **3.06** | **0** | **28.4** | **654.3** |
| | 2 | 24 | 21 | 3 | 0.32 | 0 | 117.2 | 34 |
| | 3 | 24 | 22 | 2 | 1.15 | 0 | 152.3 | 89.9 |
| Albaida | 4 | 21 | 18 | 3 | 1.23 | 0 | 159.4 | 338.7 |
| | 5 | 17 | 11 | 4 | 0.96 | 2 | 187.9 | 316.1 |
| | | **86** | **72** | **12** | **0.96** | **2** | **151.3** | **182.1** |
| | 2 | 24 | 18 | 6 | 1.20 | 0 | 323.2 | 224.1 |
| | 3 | 21 | 13 | 8 | 3.23 | 0 | 327.8 | 894.1 |
| Madrigueras | 4 | 13 | 9 | 4 | 4.19 | 0 | 251.7 | 1625.2 |
| | 5 | 5 | 4 | 1 | 3.47 | 0 | 172.0 | 2447.2 |
| | | **63** | **44** | **19** | **2.80** | **0** | **298.0** | **912.9** |
| **Total** | | **221** | **161** | **58** | **2.41** | **2** | **151.0** | **462.9** |

Table 6.3: Results of *Matheuristic 2* for the instances with known optimal solution

fact, for some of these instances, the branch-and-cut algorithm described in Ávila et al. (2017) was not capable of finding even a feasible solution in one hour of computing time, which gives an idea of the difficulty of these instances. Columns 1 and 2 of

Table 6.4 report the instance name and the number of vehicles. Column 3 shows the value of the best solution found (if any) by the exact algorithm, while column 4 reports the corresponding optimality gap. The values of the solutions provided by *Matheuristic 1* and *Matheuristic 2* and the computing times in seconds can be seen in the following four columns of the table. The last three columns present the gap obtained by the three algorithms with respect to the best solution found. Note that our algorithm *Matheuristic 2*, not only improves the solutions provided by the branch-and-cut method in many instances, but also finds a feasible solution in all of them.

| Instances | # Veh | B&C | | MH 1 | | MH 2 | | Gap Best Sol (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | UB | Gap | UB | Time | UB | Time | B&C | MH1 | MH2 |
| M3103_gdrpp | 3 | 9052 | 0.05 | 9348 | 100 | 9237 | 600 | 0 | 0.03 | 0.02 |
| M3105_gdrpp | 3 | 9454 | 0.12 | 9086 | 100 | 9086 | 600 | 0.04 | 0 | 0 |
| M3201_gdrpp | 3 | 9795 | 0.04 | 10612 | 56.8 | 10131 | 600 | 0 | 0.08 | 0.03 |
| LCGDRPP_75_3_20_1 | 4 | 9952 | 0.07 | - | 100 | 9910 | 11.2 | 0.01 | - | 0 |
| LCGDRPP_75_3_20_2 | 4 | 10684 | 0.09 | 11119 | 4.1 | 10976 | 21.3 | 0 | 0.04 | 0.03 |
| M3101_gdrpp | 4 | - | - | 9369 | 78.0 | 9307 | 600 | - | 0.01 | 0 |
| M3103_gdrpp | 4 | 9628 | 0.11 | 9954 | 100 | 9709 | 600 | 0 | 0.03 | 0.01 |
| M3105_gdrpp | 4 | - | - | 9211 | 100 | 9211 | 600 | - | 0 | 0 |
| M3111_gdrpp | 4 | 30660 | 0.12 | 29639 | 9.9 | 29637 | 600 | 0.03 | 0 | 0 |
| M3201_gdrpp | 4 | 10451 | 0.08 | 11386 | 100 | 10829 | 600 | 0 | 0.09 | 0.04 |
| M3203_gdrpp | 4 | 10145 | 0.07 | 10959 | 100 | 10344 | 600 | 0 | 0.08 | 0.02 |
| M3205_gdrpp | 4 | 11526 | 0.19 | 10824 | 100 | 10768 | 600 | 0.07 | 0.01 | 0 |
| M3211_gdrpp | 4 | 38869 | 0.11 | - | 100 | 38992 | 600 | 0 | - | 0.01 |
| M5209_gdrpp | 4 | 8834 | 0.01 | 8943 | 38.8 | 8834 | 297 | 0 | 0.01 | 0 |
| LCGDRPP_50_3_20_3 | 5 | 9648 | 0.08 | 9903 | 2.1 | 9807 | 51.3 | 0 | 0.03 | 0.02 |
| LCGDRPP_75_3_10_3 | 5 | 9858 | 0.14 | 10094 | 4.7 | 9978 | 31.5 | 0 | 0.02 | 0.01 |
| LCGDRPP_75_3_20_1 | 5 | 10615 | 0.15 | 10640 | 2.5 | 10826 | 17.9 | 0 | 0 | 0.02 |
| LCGDRPP_75_3_20_2 | 5 | 11728 | 0.17 | 12001 | 15.3 | 11871 | 7.4 | 0 | 0.02 | 0.01 |
| LCGDRPP_75_3_20_3 | 5 | - | - | 8847 | 61.4 | 8916 | 61.9 | - | 0 | 0 |
| M3101_gdrpp | 5 | - | - | 10374 | 100 | 10374 | 600 | - | 0 | 0 |
| M3103_gdrpp | 5 | - | - | 11477 | 100 | 11139 | 600 | - | 0.03 | 0 |
| M3105_gdrpp | 5 | - | - | 9461 | 65.2 | 9343 | 436 | - | 0.01 | 0 |
| M3107_gdrpp | 5 | 5368 | 0.10 | 5604 | 14 | 5234 | 214 | 0.03 | 0.07 | 0 |
| M3109_gdrpp | 5 | 10794 | 0.09 | 11272 | 21.6 | 10973 | 465 | 0 | 0.04 | 0.02 |
| M3111_gdrpp | 5 | 29686 | 0.09 | 30044 | 41.7 | 29686 | 600 | 0 | 0.01 | 0 |
| M3201_gdrpp | 5 | 12224 | 0.26 | 11947 | 100 | 11394 | 600 | 0.07 | 0.05 | 0 |
| M3203_gdrpp | 5 | 13536 | 0.42 | 11566 | 100 | 11041 | 600 | 0.23 | 0.05 | 0 |
| M3205_gdrpp | 5 | - | - | 11220 | 100 | 10638 | 600 | - | 0.05 | 0 |
| M3209_gdrpp | 5 | 11933 | 0.03 | - | 100 | 12453 | 75.5 | 0 | - | 0.04 |
| M3211_gdrpp | 5 | 40744 | 0.26 | 41837 | 65.5 | 39787 | 600 | 0.02 | 0.05 | 0 |

Table 6.4: Results for the 30 instances with unknown optimal solution

Table 6.5 summarizes the results shown in Table 6.4 obtained by the branch-and-cut and *Matheuristic 2* algorithms for the 30 instances not solved to optimality. It can be seen in the first row that *Matheuristic 2* was able to find a feasible solution for all the 30 instances while the branch-and-cut algorithm was only able to find it for 23 of them. The average UB and the average time (second and fourth rows, respectively)

refer only to these 23 instances. The number of instances for which *Matheuristic 2* obtained a better solution, 17, includes the 7 instances where the branch-and-bound did not obtain a feasible solution. As can be seen, in these 30 instances, *Matheuristic 2* clearly outperforms the branch-and-cut algorithm.

| | B&C | Matheuristic 2 |
|---|---|---|
| # of feasible solutions | 23/30 | 30/30 |
| Average UB | 14573.22 | 14413.17 |
| # of best solutions | 15/30 | 17/30 |
| Average time | 3600 | 405.3 |

Table 6.5: Overall results for the unsolved instances

In order to assess the contribution of the different constructive algorithms, we have studied the number of times each algorithm has been able to produce the best solution. Table 6.6 reports this information for all the six variants of the constructive algorithms in all the 251 instances used. It presents the percentage of best solutions found both when the route optimization is used and when it is deactivated. It can be seen that heuristics working in a parallel way perform better than those that use sequential completion. Moreover, this behavior remains true whether the optimization phase is executed or not.

| | Parallel completion | | | Sequential completion | | |
|---|---|---|---|---|---|---|
| | RI | RSBA | WSBA | RI | RSBA | WSBA |
| Without optimization | 20.92 | 39.04 | 33.07 | 13.94 | 10.56 | 13.55 |
| With optimization | 23.51 | 38.25 | 31.08 | 16.14 | 13.35 | 12.75 |

Table 6.6: Percentage of best solutions found with each constructive algorithm with and without the exact route optimization phase

| | | #Opt | #No Opt | Gap(%) | #No Sol | Time (s) | $T_{Opt}$ |
|---|---|---|---|---|---|---|---|
| With optimization | MH1 | 115 | 101 | 3.93 | 5 | 22.9 | 95.97% |
| | MH2 | 161 | 58 | 2.41 | 2 | 151.0 | 91.74% |
| Without optimization | MH1 | 72 | 138 | 3.05 | 11 | 22.9 | – |
| | MH2 | 89 | 128 | 2.29 | 4 | 151.5 | – |

Table 6.7: Impact of the exact route optimization phase on the instances with known optimal solution

Finally, in order to study the impact of the exact route optimization phase on the performance of the matheuristic, we have run a variant of MH1 and MH2 in which we have deactivated the optimization phase. These variants have been executed during the same running time used by the original versions of MH1 and MH2. Table 6.7 shows the obtained results. Last column reports the percentage of time used by the exact optimization procedure with respect to the total computing time. Although the time used in the optimization phase may seem high, if we compare, for example, the number of instances in which the optimum has been obtained or the number of those

in which no solution has been found, we can conclude that the optimization of the routes plays an important role in the performance of the matheuristic.

## 6.5 Conclusions

In this chapter we have addressed the generalization of the Close-Enough Arc Routing Problem to the case with several vehicles and maximum distance (or time) constraints. For this problem, we have proposed a matheuristic. This procedure incorporates the exact algorithm for the single vehicle case presented in Ávila et al. (2016b) in order to optimize the routes obtained. We have performed extensive computational experiments on a set of benchmark instances and the results have been compared with those obtained with the exact procedure proposed by Ávila et al. (2017). The proposed algorithm has been able to solve to optimality 117 out of 221 instances in short computing times.

# Chapter 7

# On the Distance-Constrained Close Enough Arc Routing Problem

## 7.1 Introduction

In this chapter we deepen the study of the DC-CEARP. We propose a new formulation for the DC-CEARP that combines the best features of the previously existing ones presented in Ávila et al. (2017). For this new formulation, an exhaustive study of its associated polyhedron is performed, and several different families of valid inequalities are proposed. We would like to emphasize that this contribution is not only for this problem, since many of the new inequalities presented here can be used, directly or easily adapted, in other arc routing problems. It is implemented an efficient branch-and-cut algorithm that includes the separation procedures devised for these inequalities. As before, the ideas on which some of the algorithms designed for the separation of these inequalities are based (or the algorithms themselves), can be used for similar inequalities in other problems. Extensive computational experiments are carried out to measure the contribution of each of these separation procedures. The combination with the best results is selected and compared with the results obtained by using the algorithms presented in Ávila et al. (2017). As it is shown, the designed branch-and-cut algorithm is able to solve instances with up to 140 customers, 196 vertices, 544 arcs, and 5 vehicles to optimality within two hours computing time.

This chapter is organized as follows. In Section 7.2 we describe the problem formally and present the new formulation. Several families of valid inequalities are shown in Section 7.3, while the corresponding separation methods and the branch-and-cut

algorithm are presented in Section 7.4. Computational experiments are reported in Section 7.5, and some conclusions are given in Section 7.6.

## 7.2 Problem definition and formulation

The Distance-Constrained Close Enough Arc Routing Problem, DC-CEARP, is defined as follows. Consider a strongly connected and directed graph $G = (V, A)$, where $V$ is the set of vertices, $A$ is the set of arcs, and, for each arc $(i, j) \in A$, there is a distance $d_{ij}$ associated with its traversal. Vertex 1 represents the depot. There is a fleet of $K$ identical vehicles based at the depot and a set of $L$ customers. Each customer $c \in \{1, \ldots, L\}$ has an associated set of arcs $H_c \subseteq A$ from which it can be serviced. We consider that a customer $c$ is serviced if there is a vehicle $k$ that traverses at least one arc in $H_c$. The length of the routes of the vehicles must not exceed a maximum travel distance denoted by $D_{max}$. The aim of the DC-CEARP is to find a set of $K$ routes, starting and ending at the depot, with minimum total distance and such that each customer $c = 1, \ldots, L$, is serviced and the length of each route does not exceed $D_{max}$.

In what follows, $\mathbb{K} = \{1, \ldots, K\}$ will represent the set of vehicles, $\mathbb{H} = \{1, \ldots, L\}$ the set of customers, and $A_R = H_1 \cup H_2 \cup \ldots \cup H_L$ the set of required arcs. The arcs in the set $A_{NR} = A \setminus A_R$ are called non-required arcs. Given two sets $S, T \subset V$, we define $(S : T) = \{(i, j) \in A : i \in S,\ j \in T\}$ and $(S, T) = (S : T) \cup (T : S)$. In particular, $\delta^+(S) = (S : V \setminus S)$, $\delta^-(S) = (V \setminus S : S)$ and $\delta(S) = (S, V \setminus S)$. Finally, $A(S) = \{(i, j) \in A : i, j \in S\}$ and, given a set of variables $x_{ij}$ indexed on the arcs, and given a set $F$ of arcs, $x(F) = \sum_{(i,j)\in F} x_{ij}$.

The formulation we propose here, $\mathrm{F}_{xyz}$, is based on the $\mathrm{F}_{xy+}$ and $\mathrm{F}_{xz}$ formulations presented in Ávila et al. (2017). This new formulation has more variables than $\mathrm{F}_{xy+}$ and $\mathrm{F}_{xz}$ but, as it will be seen in Section 7.5, they are useful in the exact solution of the DC-CEARP. The formulation $\mathrm{F}_{xyz}$ uses the following variables:

$x_{ij}^k$ = number of times that the vehicle $k$ traverses arc $(i, j) \in A$,

$$y_{ij}^{kc} = \begin{cases} 1, & \text{if the customer } c \text{ is serviced by vehicle } k \text{ while traversing arc } (i, j) \in A_R \\ 0, & \text{otherwise.} \end{cases}$$

$$z_c^k = \begin{cases} 1, & \text{if the customer } c \text{ is serviced by vehicle } k \\ 0, & \text{otherwise.} \end{cases}$$

The DC-CEARP can be formulated as

$$\text{Minimize} \qquad \sum_{k\in\mathbb{K}} \sum_{(i,j)\in A} d_{ij}\, x_{ij}^k$$

s.t.:

$$\sum_{(i,j)\in A} d_{ij}\, x_{ij}^k \leq D_{max} \qquad \forall k \in \mathbb{K} \tag{7.1}$$

$$x^k(\delta^+(i)) = x^k(\delta^-(i)) \qquad \forall\, i \in V,\ \forall k \in \mathbb{K} \tag{7.2}$$

$$\sum_{k\in\mathbb{K}} \sum_{(i,j)\in H_c} y_{ij}^{kc} = 1 \qquad \forall c \in \mathbb{H} \tag{7.3}$$

$$x_{ij}^k \geq y_{ij}^{kc} \qquad \forall (i,j) \in A_R,\ \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{7.4}$$

$$\sum_{(i,j)\in H_c} y_{ij}^{kc} = z_c^k \qquad \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{7.5}$$

$$x^k(\delta^+(S)) \geq z_c^k - x^k(H_c \cap A(V \setminus S)) \qquad \forall S \subset V\setminus\{1\},\ \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{7.6}$$

$$x_{ij}^k \geq 0 \ \text{ and integer} \qquad \forall (i,j) \in A,\ \forall k \in \mathbb{K} \tag{7.7}$$

$$z_c^k \in \{0,1\} \qquad \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{7.8}$$

$$y_{ij}^{kc} \in \{0,1\} \qquad \forall (i,j) \in A_R,\ \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{7.9}$$

Inequalities (7.1) limit the maximum length of each vehicle route. Constraints (7.2) are the well known symmetry equations. Inequalities (7.3) force each customer to be serviced exactly from one arc and with one vehicle, and inequalities (7.4) say that if a vehicle services a required arc then it has to traverse it. The relation between the $y_{ij}^{kc}$ and $z_c^k$ variables is given by equations (7.5). The connectivity of each route is guaranteed by inequalities (7.6). They are valid because, if vehicle $k$ does not service customer $c$, $z_c^k = 0$ and the inequality is trivially satisfied. Otherwise, if vehicle $k$ services customer $c$ by traversing an arc in $H_c \cap A(V \setminus S)$, then it does not need to traverse the cutset $\delta(S)$ and the inequality is also trivially satisfied. Only if vehicle $k$ services customer $c$ by traversing an arc not in $H_c \cap A(V \setminus S)$ (hence, traversing an arc in $\delta(S)$ or in $A(S)$), the vehicle has to traverse $\delta(S)$ and, therefore, the inequality is satisfied. Note that there is an exponential number of such inequalities. Finally, (7.7), (7.8) and (7.9) are the non-negativity and integrality constraints.

Note that the coefficients in the objective function and those in inequalities (7.1) do not necessarily have to be the same. We have set the same coefficients for the sake of simplicity and because we think of them as distances associated with a time that the routes should not exceed because they may correspond, for example, to drivers' working hours.

## 7.3 Valid inequalities

In this section we introduce some inequalities that are valid for the DC-CEARP and that will strengthen the linear relaxation of the formulation.

### 7.3.1 More connectivity inequalities

Besides the connectivity inequalities (7.6) in the formulation, involving variables $x$ and $z$, other connectivity inequalities are presented in what follows.

In Ávila et al. (2017), the following connectivity inequalities were introduced:

$$x^k(\delta^+(S)) \geq 1 - x^k(H_c \cap A(V \setminus S)) - \sum_{k' \neq k} x^{k'}(H_c), \quad \forall S \subset V \setminus \{1\}, \ \forall c \in \mathbb{H}, \ \forall k \in \mathbb{K}. \tag{7.10}$$

These inequalities ensure that, if no vehicle other than $k$ traverses the arcs in $H_c$ (thus it cannot service customer $c$), and vehicle $k$ does not traverse any arcs in $H_c \cap A(V \setminus S)$, then vehicle $k$ has to traverse the cutset $(V \setminus S, S)$ in order to service this customer. They are called *disaggregate connectivity inequalities* because they refer to a single vehicle. For each subset $\Omega \subset \mathbb{K}$ of $|\Omega| \geq 2$ vehicles, the following $\Omega$-*aggregate connectivity inequalities* are valid

$$\sum_{k \in \Omega} x^k(\delta^+(S)) \geq 1 - \sum_{k \in \Omega} x^k(H_c \cap A(V \setminus S)) - \sum_{k' \notin \Omega} x^{k'}(H_c), \ \forall S \subset V \setminus \{1\}, \ \forall c \in \mathbb{H}. \tag{7.11}$$

In the case $\Omega = \mathbb{K}$ the *aggregate connectivity inequalities* are:

$$\sum_{k \in \mathbb{K}} x^k(\delta^+(S)) \geq 1 - \sum_{k \in \mathbb{K}} x^k(H_c \cap A(V \setminus S)), \tag{7.12}$$

for any subset $S \subseteq V \setminus \{1\}$ and any customer $c \in \mathbb{H}$.

If we consider also $y_{ij}^{kc}$ variables, we have a different family of connectivity inequalities (see Ávila et al. (2017)):

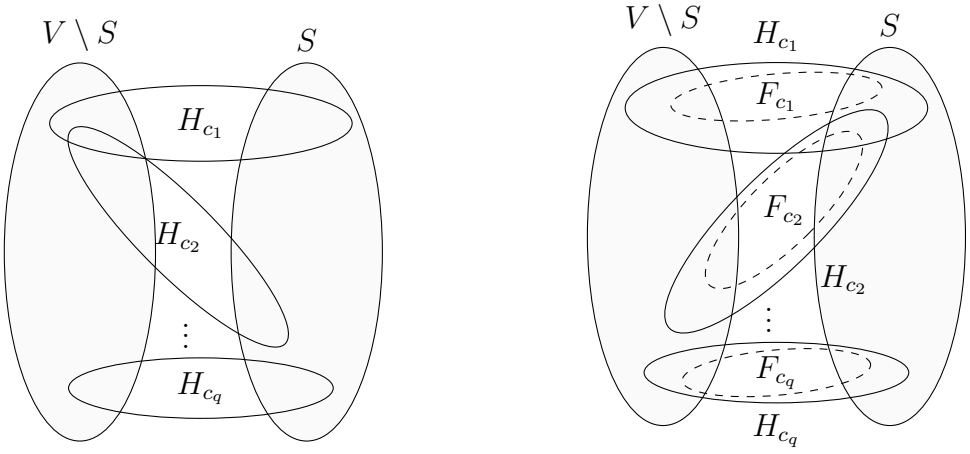
$$x^k(\delta^+(S)) \geq \sum_{(i,j) \in H_c \setminus A(V \setminus S)} y_{ij}^{kc}, \quad \forall S \subset V \setminus \{1\}, \ \forall c \in \mathbb{H}, \ \forall k \in \mathbb{K} \tag{7.13}$$

Note that, if vehicle $k$ services customer $c$ using an arc in $H_c \setminus A(V \setminus S)$, then the vehicle has to traverse $\delta(S)$.

Unlike for inequalities (7.10), the Ω-aggregate and aggregate versions of connectivity inequalities (7.6) and (7.13) are just the sum of the corresponding disaggregate inequalities and, therefore, they are dominated.

### 7.3.2 Parity inequalities

Parity inequalities are based on the fact that a vehicle crosses any cutset an even (or zero) number of times. The parity inequalities for the DC-CEARP described in what follows are different from those in other arc routing problems because they are related not only to the arcs in the cutset but also to the sets $H_c$. Four different families of *parity inequalities* were presented in Ávila et al. (2017). From them, the two stronger ones are presented in what follows.




(a) Parity Inequalities (7.14).

(b) Parity Inequalities (7.15).

Figure 7.1: Structure of the parity inequalities for the DC-CEARP

The first family uses only $x$ variables. Given a vehicle $k$, let $S \subset V$ and consider a subset of customers $F^{\mathbb{H}} = \{c_1, c_2, ..., c_q\}$, with $q \geq 3$ and odd, such that $H_{c_i} \cap H_{c_j} \cap \delta(S) = \emptyset$ and $H_{c_i} \cap \delta(S) \neq \emptyset, \forall c_i \in F^{\mathbb{H}}$ (see Figure 7.1a). In Ávila et al. (2017), it is proved that the following parity inequalities are valid for the DC-CEARP:

$$x^k(\delta(S)) \geq \sum_{i=1}^{q} \Big(1 - 2\sum_{k' \neq k} x^{k'}(H_{c_i}) - 2x^k(H_{c_i} \setminus \delta(S))\Big) + 1. \tag{7.14}$$

Note that, if no other vehicle $k' \neq k$ services customer $c_i$ (i.e., $\sum_{k' \neq k} x^{k'}(H_{c_i}) = 0$) and vehicle $k$ does not traverse any edge of customer $c_i$ that is not in the cutset (i.e., $x^k(H_{c_i} \setminus \delta(S)) = 0$), then $k$ traverses at least an arc in $H_{c_i} \cap \delta(S)$. Extending the previous argument to the $q$ customers in $\delta(S)$, vehicle $k$ has to traverse at least $q$ times $\delta(S)$ and, since $q$ is an odd number, it has to go through the cutset one more time.

The second set of inequalities use the $x_{ij}^k$ and the $y_{ij}^{kc}$ variables. Consider now the set of arc subsets $\mathcal{F} = \{F_{c_1}, F_{c_2}, ..., F_{c_q}\}$, with $q \geq 3$ and odd, satisfying $F_{c_i} \subseteq H_{c_i} \cap \delta(S)$ and $F_{c_i} \cap F_{c_j} = \emptyset, \forall c_i, c_j$ (see Figure 7.1b). Then, the following parity inequalities are valid for the DC-CEARP:

$$x^k(\delta(S)) \geq \sum_{i=1}^{q} \Big(2y^{kc_i}(F_{c_i}) - 1\Big) + 1. \tag{7.15}$$

In this case, note that if vehicle $k$ services each customer $c_i$ from an arc in $F_{c_i}$, then it has to traverse $\delta(S)$ at least $q$ times. Again, since $q$ is an odd number, the number of traversals should be at least $q + 1$.

Besides the right-hand side of the inequalities, there is a difference between the conditions satisfied by the customers in the parity inequalities above. As it is depicted in Figure 7.1, two customers $c_1$ and $c_2$ satisfying $H_{c_1} \cap H_{c_2} \cap \delta(S) \neq \emptyset$ cannot be considered for inequality (7.14), but they can for inequality (7.15) if $F_{c_1}$ and $F_{c_2}$ are chosen such that $F_{c_1} \cap F_{c_2} = \emptyset$. Nevertheless, we want to point out that the greater the sets $F_{c_i}$, the stronger inequalities (7.15). In particular, if $F_{c_i} = H_{c_i} \cap \delta(S)$, for all $c_i \in F^{\mathbb{H}}$, and $F_{c_i} \cap F_{c_j} = \emptyset$ for all $i \neq j; i, j = 1, \dots q$, we obtain the strongest inequality.

By comparing both kind of inequalities, it can be seen that none of them dominates the other in all the cases. Hence, in the branch-and-cut algorithm we will use both families of parity inequalities (7.14) and (7.15).

Finally, given a set of vehicles $\Omega = \{k_1, .., k_P\}$, $2 \leq P \leq K$, we have the following $\Omega$-*aggregate parity inequalities*:

$$\sum_{k \in \Omega} x^k(\delta^+(S)) \geq q + 1 - 2\sum_{i=1}^{q} \Big(\sum_{k' \notin \Omega} x^{k'}(H_{c_i}) + \sum_{k \in \Omega} x^k(H_{c_i} \setminus \delta(S))\Big). \tag{7.16}$$

$$\sum_{k \in \Omega} x^k(\delta^+(S)) \geq \sum_{i=1}^{q} \Big(\sum_{k \in \Omega} 2y^{kc_i}(F_{c_i}) - 1\Big) + 1. \tag{7.17}$$

It can be seen that if $\Omega = \mathbb{K}$ and $F_{c_i} = H_{c_i}$, $\forall\ c_i$ (hence $H_{c_i} \setminus \delta(S) = \emptyset$ holds), inequalities (7.16) and (7.17) reduce to the following aggregate parity inequality:

$$\sum_{k \in \mathbb{K}} x^k(\delta^+(S)) \geq q + 1.$$

### 7.3.3 K-C inequalities

The name of this family of inequalities is motivated by the number of sets into which $V$ is partitioned, which is usually denoted by $K$. To avoid confusion with the number of vehicles, in what follows we use the letter Q instead.

All the versions of the K-C inequalities are based on a structure (see Figure 7.2) defined by a partition of the set of vertices $V$ into $Q+1$ subsets, $M_0, M_1, \ldots, M_{Q-1}, M_Q$, and a set of coefficients for the arcs or edges of the graph. For each $(i,j) \in A$, we define

$$\alpha_{ij} = \begin{cases} Q-2, & \text{if } (i,j) \in (M_0, M_Q) \\ |r-s|, & \text{if } (i,j) \in (M_r, M_s),\ \{r,s\} \neq \{0,Q\} \\ 0, & \text{otherwise.} \end{cases} \tag{7.18}$$

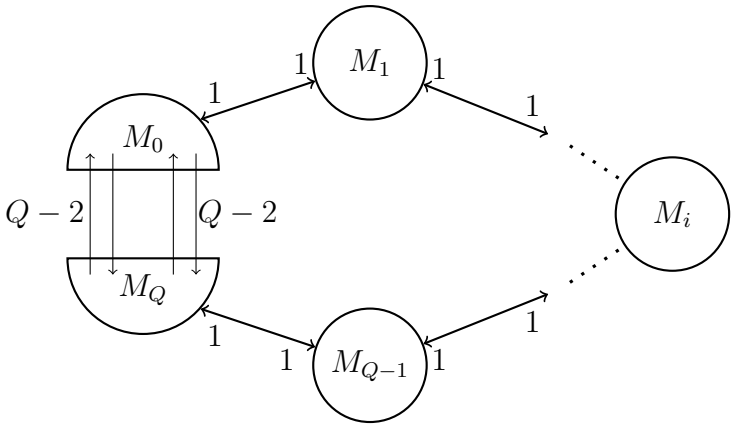



Figure 7.2: Standard K-C basic structure.

Let us call *external arcs* those joining two consecutive sets $M_r$ and $M_{r+1}$, and *internal arcs* to those joining two sets $M_r$ and $M_s$ with $|r-s| > 1$ and $\{r,s\} \neq \{0,Q\}$. Note that all the external arcs have coefficient 1, while the coefficient of an internal arc from $M_r$ to $M_s$ (not shown in Figure 7.2) is equal to the cost of the shortest path using the coefficients of the external arcs. Finally, the coefficient of the arcs in $(M_0, M_Q)$ is $Q-2$. It is known (see Corberán and Sanchis (1994) and Ávila et al. (2017)) that any vector $x \in \mathbb{Z}^{|A|}$ representing a tour traversing at least an even number $q \geq 2$ of times the arcs in $(M_0, M_Q)$, and visiting at least once each node set $M_0 \cup M_Q$, $M_1$, $\ldots$, $M_{Q-1}$, satisfies the following K-C inequality:

$$\sum_{(i,j)\in A} \alpha_{ij}\ x_{ij} \geq (Q-2)q + 2(Q-1). \tag{7.19}$$

Inequality (7.19) is valid for any ARP in which all the tours $x$ must traverse $q$ times the arcs in $(M_0, M_Q)$ and visit all the node sets $M_0 \cup M_Q$, $M_1$, $\ldots$, $M_{Q-1}$. As an

example, this is the case of the ARPs with one single vehicle when there are $q$ required arcs in $(M_0, M_Q)$ and some required arcs in all the sets $M_1, \ldots, M_{Q-1}$. In an ARP with several vehicles, such as the DC-CEARP studied here, it is usual that single vehicles are not obliged to traverse all the required arcs (or, therefore, to visit all the nodes), but only those arcs that are serviced by it. Thus, the K-C inequalities for the DC-CEARP presented in this section have the same left-hand side (LHS) as inequality (7.19), but the right-hand side (RHS) must include variables $y_{ij}^{kc}$ that define the service of a customer by a vehicle from an arc, in such a way that, when the vehicle $k$ satisfies the above conditions, the RHS of the inequality takes value $(Q-2)q + 2(Q-1)$.

### Disaggregate K-C inequalities

Consider a partition of the set of vertices $V$ into $Q$ subsets $\{M_0 \cup M_Q, M_1, \ldots, M_{Q-1}\}$, with $Q \geq 3$, and the set of coefficients $\alpha_{ij}$ given in (7.18) (see Figure 7.3).

Let us consider a family of arc subsets $\mathcal{F} = \{F_1, F_2, \ldots, F_q\}$, with $q \geq 2$ and even, satisfying (see Figure 7.3):

- $F_i \neq \emptyset \quad \forall i \in \{1, \ldots, q\}$,
- $\exists\ c_i \in \mathbb{H}$ such that $F_i \subseteq H_{c_i} \cap (M_0, M_Q), \quad \forall i \in \{1, \ldots, q\}$,
- $F_i \cap F_j = \emptyset, \quad \forall i, j \in \{1, \ldots, q\},\ i \neq j$.

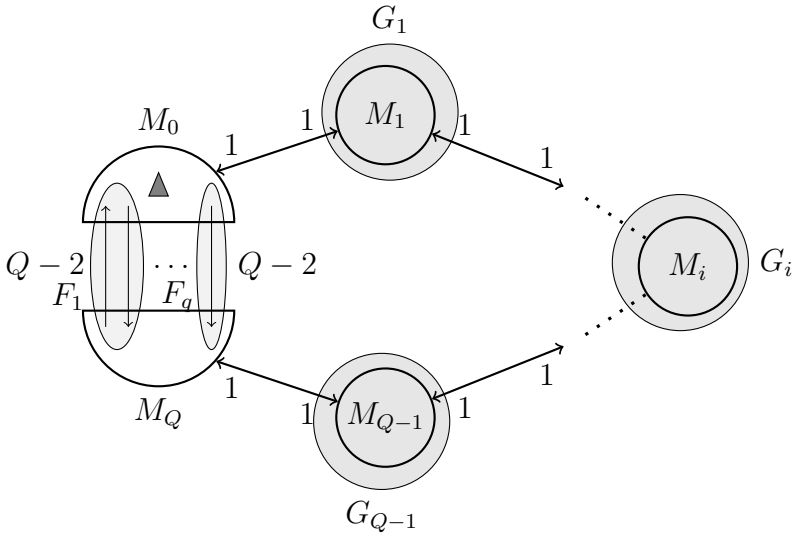



Figure 7.3: Standard disaggregate K-C inequality for the DC-CEARP. Depot is represented by a triangle.

Furthermore, assume that for each $M_j$, $j = 1, \ldots, Q-1$, either $1 \in M_j$ or the set of arcs $G_j = H_{c_j} \cap (A(M_j) \cup \delta(M_j))$ is nonempty, for some $c_j \in \mathbb{H}$. Note that we cannot assume $G_{j_1} \cap G_{j_2} = \emptyset$ because $\delta(M_{j_1})$ and $\delta(M_{j_2})$ are not necessarily disjoint sets. We

define the *disaggregate K-C inequality* associated with a vehicle $k$ as:

$$\sum_{(i,j)\in A} \alpha_{ij}\ x_{ij}^k \geq (Q-2)\sum_{i=1}^{q}\Big(2y^{kc_i}(F_i)-1\Big)+\sum_{j=1}^{Q-1}2y^{kc_j}(G_j), \tag{7.20}$$

if the depot is in $M_0 \cup M_Q$, and

$$\sum_{(i,j)\in A} \alpha_{ij}\ x_{ij}^k \geq (Q-2)\sum_{i=1}^{q}\Big(2y^{kc_i}(F_i)-1\Big)+\sum_{\substack{j=1\\ j\neq l.}}^{Q-1}2y^{kc_j}(G_j)+2, \tag{7.21}$$

if $1 \in M_l$ with $l \notin \{0, Q\}$.

**Note 3.** If $Q = 2$, then inequality (7.20) is exactly the connectivity constraint (7.6) associated with set $S = M_1$.

**Theorem 20.** *For each vehicle $k$, disaggregate K-C inequalities* (7.20) *and* (7.21) *are valid for the* DC-CEARP.

*Proof.* Let us suppose that $1 \in M_0 \cup M_Q$ (the proof for the case $1 \notin M_0 \cup M_Q$ is similar). We have to prove that all the routes $(\overline{x}^k, \overline{y}^k)$ for vehicle $k$ corresponding to DC-CEARP solutions satisfy inequality (7.20). We consider the following cases:

(a) Routes $(\overline{x}^k, \overline{y}^k)$ servicing each customer $c_i$ from a required arc in $F_i$, $i = 1, \ldots, q$, and servicing each customer $c_j$ from a required arc in $G_j$, $j = 1, \ldots, Q-1$. On the one hand, these tours $\overline{x}^k$ traverse at least $q$ times the arcs in $(M_0, M_Q)$, and visit at least once each node set $M_0 \cup M_Q$, $M_1$, ..., $M_{Q-1}$, and, hence, they satisfy (7.19):

$$\sum_{(i,j)\in A} \alpha_{ij}\overline{x}_{ij} \geq (Q-2)q + 2(Q-1).$$

Additionally, variables $\overline{y}^k$ satisfy $\overline{y}^{kc_i}(F_i) = 1$, for each $i = 1, \ldots, q$, and $\overline{y}^{kc_j}(G_j) = 1$, for each $j = 1, \ldots, Q-1$. Substituting them in the RHS of (7.20), we obtain

$$(Q-2)\sum_{i=1}^{q}(2\overline{y}^{kc_i}(F_i)-1)+\sum_{j=1}^{Q-1}2\ \overline{y}^{kc_j}(G_j) = (Q-2)q+2(Q-1).$$

Hence, $\sum_{(i,j)\in A} \alpha_{ij}\overline{x}_{ij} \geq (Q-2)\sum_{i=1}^{q}(2\overline{y}^{kc_i}(F_i)-1)+\sum_{j=1}^{Q-1}2\ \overline{y}^{kc_j}(G_j)$ holds and routes $(\overline{x}^k, \overline{y}^k)$ satisfy inequality (7.20).

(b) Routes $(\overline{x}^k, \overline{y}^k)$ servicing each customer $c_i$ from a required arc in $F_i$, $i = 1, \ldots, q$, and each customer $c_j$ from a required arc in $G_j$, $j = 1, \ldots, Q-1$, except one of them,

say $c_l$. These tours $\overline{x}^k$ traverse $q$ required arcs between $M_0$ and $M_Q$ and visit all but one the subgraphs $G(M_1), \ldots, G(M_{Q-1})$. Note that, regarding a K-C structure (see Figure 7.2), this cannot be done at an $\alpha$-cost lower than $(Q-2)q + 2(Q-1) - 2$ (otherwise, by adding two arcs connecting $M_l$ with $M_{l-1}$ we would obtain a tour satisfying (a) and (b) with $\alpha$-cost less than $(Q-2)q + 2(Q-1)$, which is impossible) and, hence, these tours satisfy

$$\sum_{(i,j)\in A} \alpha_{ij}\overline{x}_{ij} \geq (Q-2)q + 2(Q-1) - 2.$$

On the other hand, variables $\overline{y}^k$ satisfy $\overline{y}^{kc_i}(F_i) = 1$, for each $i = 1, \ldots, q$, and $\overline{y}^{kc_j}(G_j) = 1$, for all $j = 1, \ldots, Q-1$, except one of them, for which $\overline{y}^{kc_l}(G_l) = 0$. Thus, if we substitute these values in the RHS of (7.20),

$$(Q-2)\sum_{i=1}^{q}(2\overline{y}^{kc_i}(F_i)-1)+\sum_{j=1}^{Q-1} 2\,\overline{y}^{kc_j}(G_j) = (Q-2)q+2(Q-2) = (Q-2)q+2(Q-1)-2,$$

is obtained and, thus, $(\overline{x}^k, \overline{y}^k)$ satisfies (7.20).

(c) Routes $(\overline{x}^k, \overline{y}^k)$ servicing each customer $c_i$ from a required arc in $F_i$, $i = 1, \ldots, q$, and each customer $c_j$ from a required arc in $G_j$, $j = 1, \ldots, Q-1$, except a number $b$ of them ($b = 2, 3, \ldots$). As before, it can be seen that these tours $\overline{x}^k$ satisfy

$$\sum_{(i,j)\in A} \alpha_{ij}\overline{x}_{ij} \geq (Q-2)q + 2(Q-1) - 2b,$$

and the RHS of inequality (7.20) takes the value

$$(Q-2)\sum_{i=1}^{q}(2\overline{y}^{kc_i}(F_i)-1)+\sum_{j=1}^{Q-1} 2\,\overline{y}^{kc_j}(G_j) = (Q-2)q+2(Q-1-b) = (Q-2)q+2(Q-1)-2b.$$

(d) Routes $(\overline{x}^k, \overline{y}^k)$ servicing each customer $c_i$ from a required arc in $F_i$, for all $i = 1, \ldots, q$ except one of them, say $c_l$, and each customer $c_j$ from a required arc in $G_j$, $j = 1, \ldots, Q-1$. These tours $\overline{x}^k$ traverse $q-1$ (an odd number) required arcs between $M_0$ and $M_Q$ and visit all the subgraphs $G(M_1), \ldots, G(M_{Q-1})$. Regarding a K-C structure, this cannot be done at an $\alpha$-cost lower than $(Q-2)(q-2) + 2(Q-1)$ (otherwise, by adding two arcs connecting $M_0$ with $M_Q$, with $\alpha$-cost $Q-2$ each, we would obtain a tour satisfying (a) and (b) with $\alpha$-cost less than $(Q-2)q + 2(Q-1)$,

which is impossible) and, hence, these tours satisfy

$$\sum_{(i,j)\in A} \alpha_{ij}\overline{x}_{ij} \geq (Q-2)(q-2) + 2(Q-1).$$

Moreover, variables $\overline{y}^k$ satisfy $\overline{y}^{kc_i}(F_i) = 1$, for each $i = 1, \ldots, q$ except one of them, for which $\overline{y}^{kc_l}(F_l) = 0$, and $\overline{y}^{kc_j}(G_j) = 1$, for all $j = 1, \ldots, Q-1$. Thus, after substituting these values in the RHS of (7.20) we obtain

$$(Q-2)\sum_{i=1}^{q}(2\overline{y}^{kc_i}(F_i) - 1) + \sum_{j=1}^{Q-1} 2\ \overline{y}^{kc_j}(G_j) = (Q-2)(q-1-1) + 2(Q-1),$$

and $(\overline{x}^k, \overline{y}^k)$ satisfies (7.20).

(e) Routes $(\overline{x}^k, \overline{y}^k)$ servicing each customer $c_i$ from a required arc in $F_i$, for all $i = 1, \ldots, q$ except two of them, and each customer $c_j$ from a required arc in $G_j$, $j = 1, \ldots, Q-1$. These tours $\overline{x}^k$ traverse $q-2$ (an even number) required arcs between $M_0$ and $M_Q$ and visit all the subgraphs $G(M_1), \ldots, G(M_{Q-1})$, so they satisfy

$$\sum_{(i,j)\in A} \alpha_{ij}\overline{x}_{ij} \geq (Q-2)(q-2) + 2(Q-1).$$

Variables $\overline{y}^k$ satisfy $\overline{y}^{kc_i}(F_i) = 1$, for each $i = 1, \ldots, q$ except two of them, for which $\overline{y}^{kc_l}(F_l) = 0$, and $\overline{y}^{kc_j}(G_j) = 1$, for all $j = 1, \ldots, Q-1$, and the RHS of inequalities (7.20) takes the value

$$\begin{aligned}(Q-2)\sum_{i=1}^{q}(2\overline{y}^{kc_i}(F_i) - 1) + \sum_{j=1}^{Q-1} 2\ \overline{y}^{kc_j}(G_j) &= (Q-2)(q-2-1-1) + 2(Q-1) \\ &< (Q-2)(q-2) + 2(Q-1),\end{aligned}$$

and $(\overline{x}^k, \overline{y}^k)$ satisfies (7.20).

(f) Routes $(\overline{x}^k, \overline{y}^k)$ servicing each customer $c_i$ from a required arc in $F_i$, for all $i = 1, \ldots, q$ except three, four, ... of them, and each customer $c_j$ from a required arc in $G_j$, $j = 1, \ldots, Q-1$. By using a similar reasoning, it can be proved that they satisfy inequality (7.20).

(g) Routes $(\overline{x}^k, \overline{y}^k)$ similar to those in the cases (d), (e) and (f) but where each customer $c_j$ is serviced from a required arc in $G_j$, $j = 1, \ldots, Q-1$, except a number $b$ of them ($b = 1, 2, \ldots$). It can be seen that both the term $\sum \alpha_{ij}\overline{x}_{ij}$ and the RHS of inequality (7.20) decrease in $2b$ units, thus satisfying inequality (7.20). □

### Ω-aggregate K-C inequalities

Here we present the K-C inequalities associated with any subset of vehicles $\Omega \subseteq \mathbb{K}$. Note that, inequality (7.20) can be written as

$$\sum_{(i,j)\in A} \alpha_{ij}\ x_{ij}^k - (Q-2)\sum_{i=1}^{q}\Big(2y^{kc_i}(F_i)\Big) - \sum_{j=1}^{Q-1} 2y^{kc_j}(G_j) \geq -(Q-2)q, \qquad (7.22)$$

where the values for coefficients $\alpha_{ij}$ are given in (7.18).

If we consider a subset of vehicles $\Omega \subseteq \mathbb{K}$ and we add the $|\Omega|$ corresponding disaggregate K-C inequalities we obtain an inequality that is obviously valid for the DC-CEARP, but it is not interesting for the problem, since it is dominated:

$$\sum_{k\in\Omega}\sum_{(i,j)\in A} \alpha_{ij}\ x_{ij}^k - (Q-2)\sum_{k\in\Omega}\sum_{i=1}^{q}\Big(2y^{kc_i}(F_i)\Big) - \sum_{k\in\Omega}\sum_{j=1}^{Q-1} 2y^{kc_j}(G_j) \geq -|\Omega|(Q-2)q$$

However, by changing the RHS from $-|\Omega|(Q-2)q$ to $-(Q-2)q$, we obtain new and stronger inequalities (except when RHS=0, i.e., when $Q=3$, $q=2$ and the depot is not in $M_0 \cup M_Q$). Specifically, given a partition $\{M_0 \cup M_Q, M_1, \ldots, M_{Q-1}\}$, $Q \geq 3$, with the corresponding set of coefficients $\alpha$, a set $\mathcal{F} = \{F_1, F_2, \ldots, F_q\}$ ($q \geq 2$ and even) and some sets $G_j$ as above, and given a subset of vehicles $\Omega$, we define the *Ω-aggregate K-C inequality* as

$$\sum_{k\in\Omega}\sum_{(i,j)\in A} \alpha_{ij}\ x_{ij}^k \geq (Q-2)\sum_{i=1}^{q}\Big(\sum_{k\in\Omega} 2y^{kc_i}(F_i) - 1\Big) + \sum_{j=1}^{Q-1}\sum_{k\in\Omega} 2y^{kc_j}(G_j) \qquad (7.23)$$

if the depot $1 \in M_0 \cup M_Q$, and

$$\sum_{k\in\Omega}\sum_{(i,j)\in A} \alpha_{ij}\ x_{ij}^k \geq (Q-2)\sum_{i=1}^{q}\Big(\sum_{k\in\Omega} 2y^{kc_i}(F_i) - 1\Big) + \sum_{\substack{j=1\\ j\neq l.}}^{Q-1}\sum_{k\in\Omega} 2y^{kc_j}(G_j) + 2 \qquad (7.24)$$

if $1 \in M_l$, with $l \notin \{0, Q\}$.

**Theorem 21.** *Given a set of vehicles $\Omega \subseteq \mathbb{K}$, the Ω-aggregate K-C inequalities (7.23) and (7.24) are valid for the* DC-CEARP.

*Proof.* Again, let us suppose that $1 \in M_0 \cup M_Q$ (the proof for the case $1 \notin M_0 \cup M_Q$ is similar). We have to prove that every DC-CEARP solution satisfies inequality (7.23). Let $(\overline{x}^1, \overline{y}^1, \ldots, \overline{x}^K, \overline{y}^K)$ be a DC-CEARP solution. Then, $\sum_{k\in\Omega} \overline{x}^k$ is a tour on the

arcs of $G$ since it represents a connected and even graph. On the other hand, for each $i = 1, \ldots, q$, the sum $\sum_{k\in\Omega} \overline{y}^{kc_i}(F_i)$, is a binary value indicating if any of the vehicles in $\Omega$ services the customer $c_i$ from an arc in $F_i$ (see inequalities (7.3)). In the same way, for each $j = 1, \ldots, Q-1$, the sum $\sum_{k\in\Omega} \overline{y}^{kc_j}(G_j)$ is a binary value indicating if any of the vehicles in $\Omega$ services the customer $c_j$ from an arc in $G_j$. Hence, a similar reasoning to that of the proof of Theorem 20, but replacing $(\overline{x}^k, \overline{y}^k)$ by $(\sum_{k\in\Omega} \overline{x}^k, \sum_{k\in\Omega} \overline{y}^k)$, concludes that the following inequality, which can be rewritten as inequality (7.23), is satisfied:

$$\sum_{(i,j)\in A} \alpha_{ij} \sum_{k\in\Omega} \overline{x}^k_{ij} \geq (Q-2) \sum_{i=1}^{q} \left(2 \sum_{k\in\Omega} \overline{y}^{kc_i}(F_i) - 1\right) + \sum_{j=1}^{Q-1} 2 \sum_{k\in\Omega} \overline{y}^{kc_j}(G_j).$$

□

### 7.3.4 K-C$_{02}$ inequalities

K-C$_{02}$ inequalities are a variant of the K-C inequalities that take into account the asymmetry of the costs associated with the direction of traversal. In some ARPs the K-C$_{02}$ inequalities are dominated by the standard K-C inequalities. This is not the case for the DC-CEARP. For example, consider the fractional DC-CEARP solution $(\overline{x}^k, \overline{y}^k)$ depicted in Figure 7.4. It can be seen that this solution satisfies all the connectivity

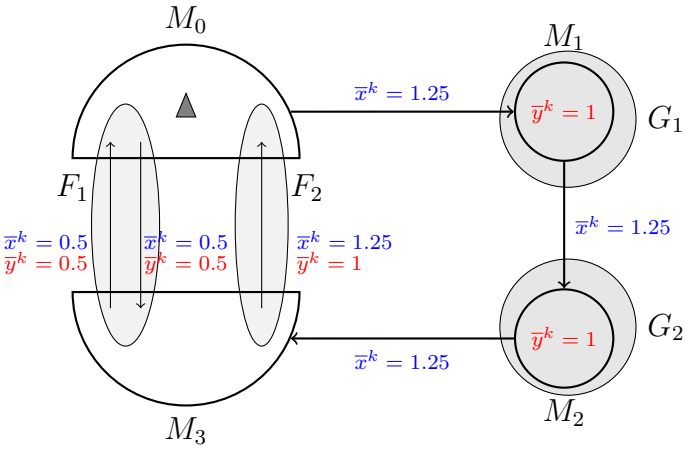



Figure 7.4: A fractional solution for vehicle $k$ not cut off by a disaggregate K-C inequality

inequalities (7.6) and it also satisfies the K-C inequality (7.20) corresponding to this structure:

$$\sum_{(i,j)\in A} \alpha_{ij}\ \overline{x}^k_{ij} = (0.5 + 0.5 + 1.25) + (1.25 + 1.25 + 1.25) = 6, \quad \text{and}$$

$$(Q-2)\sum_{i=1}^{q}\Big(2\overline{y}^{kc_i}(F_i)-1\Big)+\sum_{i=1}^{q}2\overline{y}^{kc_j}(G_j)=\Big(2(0.5+0.5)-1\Big)+\Big(2\times1-1\Big)+2\times1+2\times1=6.$$

We will see that the disaggregate K-$C_{02}$ inequalities that we describe in what follows do cut off this solution.

### Disaggregate K-$C_{02}$ inequalities

Consider a partition of the set of vertices $V$ into $Q$ subsets $\{M_0\cup M_Q, M_1,\ldots,M_{Q-1}\}$, with $Q\geq 2$, and the following set of coefficients (see Figure 7.5). For each $(i,j)\in A$,

$$\beta_{ij}=\begin{cases} Q-1, & \text{if } (i,j)\in(M_0,M_Q)\\ s-1, & \text{if } (i,j)\in(M_0:M_s),\ 1\leq s\leq Q-1\\ s+1, & \text{if } (i,j)\in(M_s:M_0),\ 1\leq s\leq Q-1\\ |r-s|, & \text{if } (i,j)\in(M_r,M_s),\ 1\leq r,s\leq Q\\ 0, & \text{otherwise.}\end{cases}$$

Let us also consider a family of arc subsets $\mathcal{F}=\{F_1,F_2,\ldots,F_q\}$, and the arc sets $G_j$ satisfying the same conditions as for the K-C inequalities. Note that now we have $Q\geq 2$ (see Note 4 below). We define the *disaggregate K-$C_{02}$ inequalities* associated with a vehicle $k$ as:

$$\sum_{(i,j)\in A}\beta_{ij}\ x_{ij}^k\geq(Q-1)\sum_{i=1}^{q}\Big(2y^{kc_i}(F_i)-1\Big)+\sum_{j=1}^{Q-1}2y^{kc_j}(G_j),\qquad(7.25)$$

if the depot is in $M_0\cup M_Q$, and

$$\sum_{(i,j)\in A}\beta_{ij}\ x_{ij}^k\geq(Q-1)\sum_{i=1}^{q}\Big(2y^{kc_i}(F_i)-1\Big)+\sum_{\substack{j=1\\ j\neq l.}}^{Q-1}2y^{kc_j}(G_j)+2,\qquad(7.26)$$

if $1\in M_l$ with $l\notin\{0,Q\}$.

**Theorem 22.** *For each vehicle $k$, K-$C_{02}$ inequalities* (7.25) *and* (7.26) *are valid for the* DC-CEARP.

*Proof.* The proof is similar to that of Theorem 20 and is omitted here for the sake of brevity. □

Let us now check that the K-$C_{02}$ inequality cuts off the fractional solution $(\overline{x}^k,\overline{y}^k)$ depicted in Figure 7.4:

$$\sum_{(i,j)\in A}\beta_{ij}\ \overline{x}_{ij}^k=2(0.5+0.5+1.25)+0\times1.25+1\times1.25+1\times1.25=7,\quad\text{while}$$

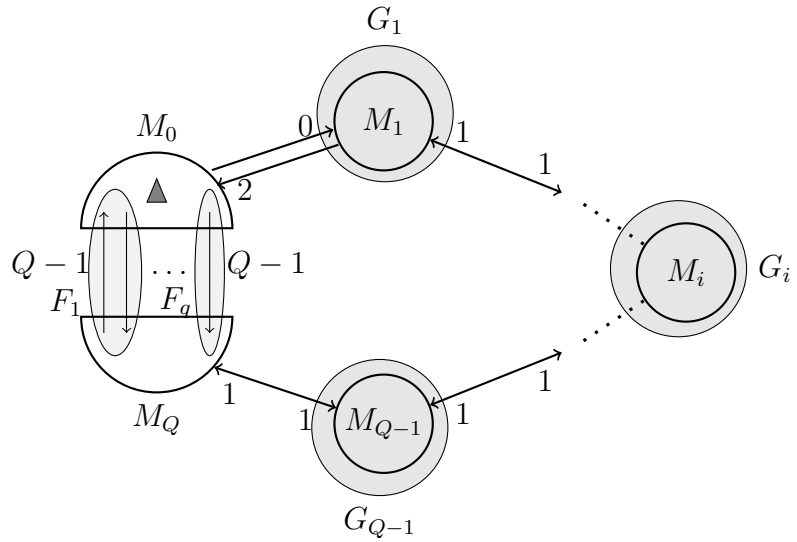



Figure 7.5: Disaggregate K-$C_{02}$ inequalities for the DC-CEARP

$$(Q-1)\sum_{i=1}^{q}\Big(2\overline{y}^{kc_i}(F_i)-1\Big)+\sum_{i=1}^{q}2\overline{y}^{kc_j}(G_j)=2\Big(2(0.5+0.5)-1+2\times1-1)\Big)+2\times1+2\times1=8.$$

**Note 4.** Unlike the standard K-C inequalities, the K-$C_{02}$ inequalities with $Q=2$ are not equivalent to any other known inequality.

### $\Omega$-aggregate K-$C_{02}$ inequalities

Given a partition $\{M_0\cup M_Q, M_1,\ldots,M_{Q-1}\}$, $Q\geq 2$, with the corresponding set of coefficients $\beta$, a set $\mathcal{F}=\{F_1,F_2,\ldots,F_q\}$ ($q\geq 2$ and even), some sets $G_j$ as above, and given a subset of vehicles $\Omega\subseteq\mathbb{K}$, we define the $\Omega$-*aggregate K-$C_{02}$ inequality* as

$$\sum_{k\in\Omega}\sum_{(i,j)\in A}\beta_{ij}\ x_{ij}^k\geq(Q-1)\sum_{i=1}^{q}\Big(\sum_{k\in\Omega}2y^{kc_i}(F_i)-1\Big)+\sum_{j=1}^{Q-1}\sum_{k\in\Omega}2y^{kc_j}(G_j),\qquad(7.27)$$

if the depot $1\in M_0\cup M_Q$, and

$$\sum_{k\in\Omega}\sum_{(i,j)\in A}\beta_{ij}\ x_{ij}^k\geq(Q-1)\sum_{i=1}^{q}\Big(\sum_{k\in\Omega}2y^{kc_i}(F_i)-1\Big)+\sum_{\substack{j=1\\ j\neq l.}}^{Q-1}\sum_{k\in\Omega}2y^{kc_j}(G_j)+2,\quad(7.28)$$

if $1\in M_l$, with $l\notin\{0,Q\}$.

**Theorem 23.** *Given a set of vehicles $\Omega\subseteq\mathbb{K}$, the $\Omega$-aggregate K-C inequalities* (7.27) *and* (7.28) *are valid for the* DC-CEARP.

*Proof.* The proof is similar to that of Theorem 21 and is omitted here for the sake of brevity. □

### 7.3.5 Path-Bridge inequalities

Path-Bridge inequalities are a generalization of the K-C inequalities introduced in Letchford (1997) for the undirected General Routing Problem and are inspired by the path inequalities introduced in Cornuèjols et al. (1985) for the Graphical Traveling Salesman Problem.

As K-C, Path-Bridge inequalities try that connectivity and parity conditions are satisfied simultaneously on a given partition of the vertex set $V$. They are based on a structure (see Figure 7.6) with two sets $M_0, M_Z$, a number $P \geq 1$ of 'paths' between $M_0$ and $M_Z$, and a number $B \geq 0$ of required arcs in $(M_0, M_Z)$ forming the 'bridge', with $P+B \geq 3$ being an odd number. It can be noted that K-C inequalities described in Section 7.3.3 are a particular case of the Path-Bridge inequalities when $P = 1$ and $B \geq 2$ and even.

#### Disaggregate path-bridge inequalities

Given two integers $P \geq 1$, $B \geq 0$ such that $P + B \geq 3$ is an odd number, consider the partition of $V$ into the subsets $\{M_0, M_Z, \{M_r^t\}_{r=1,\ldots,n_t}^{t=1,\ldots,P}\}$, where $n_1, n_2, \ldots, n_t$ are integer numbers, $n_i \geq 2$, and consider the following coefficients (see Figure 7.6). For each $(i,j) \in A$,

$$\alpha_{i,j} = \begin{cases} 1, & \text{if } (i,j) \in (M_0, M_Z) \\ \frac{|r-s|}{n_t-1}, & \text{if } (i,j) \in (M_r^t, M_s^t), \begin{cases} t \in \{1,\ldots,P\}, \\ r,s \in \{0,1,\ldots,n_t+1\} \end{cases} \\ \frac{1}{n_t-1} + \frac{1}{n_u-1} + \left|\frac{r-1}{n_t-1} - \frac{s-1}{n_u-1}\right|, & \text{if } (i,j) \in (M_r^t, M_s^u), \begin{cases} t \neq u, \\ r \in \{1,\ldots,n_t\}, \\ s \in \{1,\ldots,n_u\} \end{cases} \\ 0, & \text{otherwise.} \end{cases}$$

Let us consider a family of arc subsets $\mathcal{F} = \{F_1, F_2, \ldots, F_B\}$ satisfying:

- $F_i \neq \emptyset \quad \forall i \in \{1,\ldots,B\}$,
- $\exists\ c_i \in \mathbb{H}$ such that $F_i \subseteq H_{c_i} \cap (M_0, M_Z) \quad \forall i \in \{1,\ldots,B\}$,
- $F_i \cap F_j = \emptyset, \quad \forall i \neq j \in \{1,\ldots,B\}$.

Furthermore, assume that for each $M_r^t$, $t = 1,\ldots,P,\ r = 1,\ldots,n_t$, either $1 \in M_r^t$ or it exists a set of arcs $G_r^t$ such that $\emptyset \neq G_r^t \subseteq H_{c_j} \cap A_R(M_r^t \cup \delta(M_r^t))$ for a $c_j \in \mathbb{H}$. We define the *disaggregate Path-Bridge inequality* associated with vehicle $k$ as:

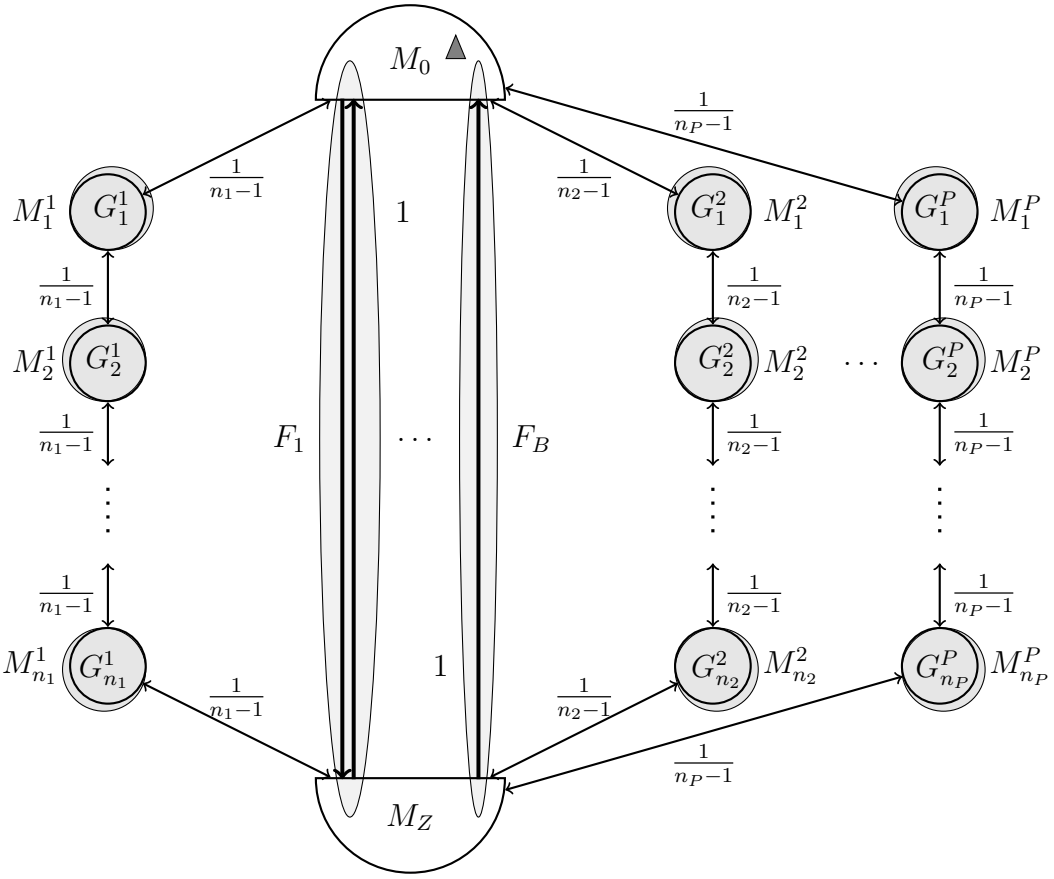



Figure 7.6: Standard Path-Bridge for the DC-CEARP

$$\sum_{(i,j)\in A} \alpha_{ij}\, x_{ij}^k \geq \sum_{i=1}^{B} \Big(2y^{kc_i}(F_i) - 1\Big) + \sum_{t=1}^{P}\sum_{r=1}^{n_t} \frac{2y^{kc_j}(G_r^t)}{n_t - 1} - P + 1, \tag{7.29}$$

if the depot $1 \in M_0 \cup M_Z$, and

$$\sum_{(i,j)\in A} \alpha_{ij}\, x_{ij}^k \geq \sum_{i=1}^{B} \Big(2y^{kc_i}(F_i) - 1\Big) + \sum_{\substack{t=1\\ t\neq t_0}}^{P}\sum_{r=1}^{n_t} \frac{2y^{kc_j}(G_r^t)}{n_t - 1} + \sum_{\substack{r=1\\ r\neq r_0}}^{n_{t_0}} \frac{2y^{kc_j}(G_r^{t_0})}{n_{t_0} - 1} + \frac{2}{(n_{t_0} - 1)} - P + 1, \tag{7.30}$$

if the depot $1 \in M_{r_0}^{t_0}$ (different from $M_0$ and $M_Z$).

**Theorem 24.** *For each vehicle $k$, disaggregate Path-Bridge inequalities* (7.29) *and* (7.30) *are valid for the* DC-CEARP.

*Proof.* Let us suppose that $1 \in M_0 \cup M_Z$ (the proof for the case $1 \notin M_0 \cup M_Z$ is similar). We have to prove that all the single routes $(\overline{x}^k, \overline{y}^k)$ for vehicle $k \in \mathbb{K}$ corresponding to DC-CEARP solutions satisfy inequality (7.29). We consider the following cases:

(a) Routes $(\overline{x}^k, \overline{y}^k)$ servicing each customer $c_i$ from an arc in $F_i$, $i = 1, \ldots, B$, and servicing each customer $c_j$ from an arc in $G_r^t$, $t = 1, \ldots, P$ and $r = 1, \ldots, n_t$. On the

one hand, these tours $\overline{x}^k$ traverse at least $B$ times the arcs in $(M_0, M_Z)$, and visit at least once all the node sets $M_0 \cup M_Z$ and $M_r^t$. It can be seen (see Corberán et al. (2001)) that these tours satisfy:

$$\sum_{(i,j)\in A} \alpha_{ij}\overline{x}_{ij}^k \geq B + \sum_{t=1}^{P} \frac{2n_t}{n_t - 1} - P + 1.$$

On the other hand, variables $\overline{y}^k$ satisfy $\overline{y}^{kc_i}(F_i) = 1$, for each $i = 1, \ldots, B$, and $\overline{y}^{kc_j}(G_r^t) = 1$, for each $t = 1, \ldots, P$ and $r = 1, \ldots, n_t$. Substituting these values in the RHS of (7.29) we obtain

$$\sum_{i=1}^{B} \Big(2\overline{y}^{kc_i}(F_i) - 1\Big) + \sum_{t=1}^{P}\sum_{j=1}^{n_t} \frac{2\overline{y}^{kc_j}(G_j^t)}{n_t - 1} - P + 1 = B + \sum_{t=1}^{P} \frac{2n_t}{n_t - 1} - P + 1.$$

Hence, $\sum_{(i,j)\in A} \alpha_{ij}\overline{x}_{ij}^k \geq \sum_{i=1}^{B} \Big(2\overline{y}^{kc_i}(F_i)-1\Big)+\sum_{t=1}^{P}\sum_{j=1}^{n_t} \frac{2\overline{y}^{kc_j}(G_j^t)}{n_t - 1}-P+1$ holds, and routes

$(\overline{x}^k, \overline{y}^k)$ satisfy inequality (7.29).

(b) Routes $(\overline{x}^k, \overline{y}^k)$ servicing each customer $c_i$ from a required arc in $F_i$, $i = 1, \ldots, B$, and servicing each customer $c_j$, except one of them ($H_{c_l} \in G_{r_0}^{t_0}$), from a required arc in $G_r^t$, $t = 1, \ldots, P$, $r = 1, \ldots, n_t$. These tours $\overline{x}^k$ traverse $B$ times some required arcs between $M_0$ and $M_Z$ and visit all the sets $M_r^t$ except the set $M_{r_0}^{t_0}$. Note that this cannot be done with an $\alpha$-cost lower than $B + \sum_{t=1}^{P} \frac{2n_t}{n_t-1} - P + 1 - \frac{2}{n_{t_0}-1}$. Otherwise, by adding two arcs connecting $M_{r_0}^{t_0}$ with $M_{r_0-1}^{t_0}$, with $\alpha-$cost $\frac{2}{n_{t_0}-1}$, we would obtain a tour, traversing at least $B$ times the arcs in $(M_0, M_Z)$ and visiting all the node sets $M_0 \cup M_Z$ and $M_r^t$, with $\alpha$-cost less than $B + \sum_{t=1}^{P} \frac{2n_t}{n_t-1} - P + 1$, which is impossible. Hence, these tours $\overline{x}^k$ satisfy

$$\sum_{(i,j)\in A} \alpha_{ij}\overline{x}_{ij}^k \geq B + \sum_{t=1}^{P} \frac{2n_t}{n_t - 1} - P + 1 - \frac{2}{n_{t_0} - 1}.$$

Moreover, variables $\overline{y}^k$ satisfy $\overline{y}^{kc_i}(F_i) = 1$, for each $i = 1, \ldots, B$, and $\overline{y}^{kc_j}(G_r^t) = 1$, for all $t = 1, \ldots, P$ and $r = 1, \ldots, n_t$, except one of them, for which $\overline{y}^{kc_l}(G_{r_0}^{t_0}) = 0$.

Thus, the RHS of inequalities (7.29) takes the value

$$(2B - B) + \sum_{\substack{t=1 \\ t \neq t_0}}^{P} \frac{2n_t}{n_t - 1} + \frac{2(n_{t_0} - 1)}{n_{t_0} - 1} - P + 1 = B + \sum_{t=1}^{P} \frac{2n_t}{n_t - 1} - P + 1 - \frac{2}{n_{t_0} - 1}$$

and, hence, the routes $(\overline{x}^k, \overline{y}^k)$ satisfy (7.29).

(c) Routes $(\overline{x}^k, \overline{y}^k)$ servicing each customer $c_i$ from a required arc in $F_i$, for all $i = 1, \ldots, B$ except one of them, say $c_l$, and each customer $c_j$ from a required arc in $G_r^t$, $t = 1, \ldots, P$, $r = 1, \ldots, n_t$.

Tours $\overline{x}^k$ traverse $B - 1$ required arcs between $M_0$ and $M_Z$ and visit all the sets $M_r^t$. Considering that $P + B - 1$ is an even number, this cannot be done with a $\alpha$-cost lower than $B + \sum_{t=1}^{P} \frac{2n_t}{n_t - 1} - P + 1 - 2$. Otherwise, by adding two arcs connecting $M_0$ with $M_Z$, with $\alpha$-cost 1 each, we would obtain a tour, traversing at least $B$ times the arcs in $(M_0, M_Z)$ and visiting all the node sets $M_0 \cup M_Z$ and $M_r^t$, with $\alpha$-cost less than $B + \sum_{t=1}^{P} \frac{2n_t}{n_t - 1} - P + 1$, which is impossible. Hence, these tours $\overline{x}^k$ satisfy

$$\sum_{(i,j) \in A} \alpha_{ij} \overline{x}_{ij} \geq = B + \sum_{t=1}^{P} \frac{2n_t}{n_t - 1} - P + 1 - 2.$$

Additionally, variables $\overline{y}^k$ satisfy $\overline{y}^{kc_i}(F_i) = 1$, for each $i = 1, \ldots, B$ except one of them, for which $\overline{y}^{kc_l}(F_l) = 0$, and $\overline{y}^{kc_j}(G_r^t) = 1$, for all $t = 1, \ldots, P$, $r = 1, \ldots, n_t$. Thus, after substituting them in the RHS of (7.29), we obtain

$$2(B - 1) - B + \sum_{t=1}^{P} \frac{2n_t}{n_t - 1} - P + 1 = B + \sum_{t=1}^{P} \frac{2n_t}{n_t - 1} - P + 1 - 2,$$

and the routes $(\overline{x}^k, \overline{y}^k)$ satisfy (7.29).

(d) In a similar way, it can be seen that inequalities (7.29) are satisfied by all the routes $(\overline{x}^k, \overline{y}^k)$ servicing any other number of customers $c_i$ and $c_j$. □

If we multiply the Path-Bridge inequalities (7.29) and (7.30) by $\prod_{t=1}^{P}(n_t - 1)$, all the coefficients become integer. When $P = 2$ (and, hence, $B$ is an odd number), the inequality is called *2-Path-Bridge inequality* and can be written as:

$$\sum_{(i,j)\in A}(n_1-1)(n_2-1)\alpha_{ij}\ x_{ij}^k \geq \sum_{i=1}^{B} 2(n_1-1)(n_2-1)y^{kc_i}(F_i)+$$

$$+\sum_{j=1}^{n_1} 2(n_2-1)y^{kc_j}(G_j^1)+\sum_{j=1}^{n_2} 2(n_1-1)y^{kc_j}(G_j^2)-(B+1)(n_1-1)(n_2-1), \quad (7.31)$$

when the depot $1 \in M_0 \cup M_Z$. If the depot is, for example, in a node $j_0$ of the path $t=1$ ($1 \in M_{j_0}^1$), the resulting inequality is

$$\sum_{(i,j)\in A}(n_1-1)(n_2-1)\alpha_{ij}\ x_{ij}^k \geq \sum_{i=1}^{B} 2(n_1-1)(n_2-1)y^{kc_i}(F_i)+2(n_2-1)$$

$$+\sum_{\substack{j=1\\ j\neq j_0}}^{n_1} 2(n_2-1)y^{kc_j}(G_j^1)+\sum_{j=1}^{n_2} 2(n_1-1)y^{kc_j}(G_j^2)-(B+1)(n_1-1)(n_2-1). \quad (7.32)$$

### Ω-aggregate Path-Bridge inequalities

Given $P, B$, a partition $\{M_0, M_Z, \{M_r^t\}_{r=1,\dots,n_t}^{t=1,\dots,P}\}$, its corresponding set of coefficients $\alpha_{i,j}$, and the families of arcs $F_i$ and $G_r^t$, as in the previous section, we define the Ω-*aggregate Path-Bridge inequality* associated with a subset $\Omega \subseteq \mathbb{K}$ of vehicles as:

$$\sum_{k\in\Omega}\sum_{(i,j)\in A}\alpha_{ij}\ x_{ij}^k \geq \sum_{i=1}^{B}\Big(\sum_{k\in\Omega}2y^{kc_i}(F_i)-1\Big)+\sum_{k\in\Omega}\sum_{s=1}^{P}\Big(\sum_{q=1}^{n_t}\frac{2y^{kc_j}(G_r^t)}{n_t-1}\Big)-P+1, \quad (7.33)$$

if the depot $1 \in M_0 \cup M_Z$, and

$$\sum_{k\in\Omega}\sum_{(i,j)\in A}\alpha_{ij}\ x_{ij}^k \geq$$

$$\geq \sum_{i=1}^{B}\Big(\sum_{k\in\Omega}2y^{kc_i}(F_i)-1\Big)+\sum_{k\in\Omega}\sum_{\substack{t=1\\ t\neq t_0}}^{P}\sum_{r=1}^{n_t}\frac{2y^{kc_j}(G_r^t)}{n_t-1}+\sum_{k\in\Omega}\sum_{\substack{r=1\\ r\neq r_0}}^{n_{t_0}}\frac{2y^{kc_j}(G_r^{t_0})}{n_{t_0}-1}+\frac{2}{(n_{t_0}-1)}-P+1, \quad (7.34)$$

if the depot $1 \in M_{r_0}^{t_0}$ (different from $M_0$ and $M_Z$).

**Theorem 25.** *Given a set of vehicles $\Omega \subseteq \mathbb{K}$, the Ω-aggregate Path-Bridge inequalities* (7.33) *and* (7.34) *are valid for the* DC-CEARP.

*Proof.* The proof is similar to that of Theorem 21 and is omitted here for the sake of brevity. □

**Note 5** (Path-Bridge$_{02}$ inequalities)**.** An asymmetric version of Path-Bridge inequalities, called Path-Bridge$_{02}$ inequalities, is proposed in Corberán et al. (2003) for the Mixed General Routing Problem. Inequalities based on the same idea can also be proposed for the DC-CEARP. However, not all their coefficients can be easily determined, since the coefficients of the variables associated with arcs between nodes of different paths must be computed by sequential lifting for each particular Path-Bridge structure. This process is involved and, in addition, the obtained coefficients depend on the ordering in which arcs are considered. For this reason, its separation has never been implemented and, therefore, these inequalities are not studied here.

### 7.3.6 Max-distance constraints

In the DC-CEARP, the length of each route cannot exceed the maximum distance $D_{max}$. Based on this constraint, in this section we present several sets of inequalities that we call *max-distance inequalities*.

Let $F^{\mathbb{H}} \subseteq \mathbb{H}$ be a subset of customers. Consider the Close-Enough Arc Routing Problem (which considers only one vehicle), defined on graph $G$ and with set of customers $F^{\mathbb{H}}$. Let CEARP($F^{\mathbb{H}}$) be its optimal value (or a lower bound of it). If CEARP($F^{\mathbb{H}}$)$> D_{max}$, then the inequalities

$$z_c^k(F^{\mathbb{H}}) \leq |F^{\mathbb{H}}| - 1, \quad \forall k \in \mathbb{K} \tag{7.35}$$

are valid for the DC-CEARP, because a single vehicle cannot service all the customers in $F^{\mathbb{H}}$.

On the other hand, if $S$ is the set of vertices incident with the arcs in $\cup_{c \in F^{\mathbb{H}}} H_c$ and $1 \notin S$, then at least two different vehicles have to enter $S$, and the following inequality is also valid for the DC-CEARP

$$\sum_{k \in \mathbb{K}} x^k(\delta^-(S)) \geq 2. \tag{7.36}$$

However, a DC-CEARP solution in which a vehicle enters $S$ twice, but no other vehicle does, satisfies (7.36). In order to force two different vehicles to enter $S$, the following valid inequalities can be used

$$\sum_{k \neq k'} x^k(\delta^-(S)) \geq 1, \quad \forall k' \in \mathbb{K}. \tag{7.37}$$

The above inequalities, which were proposed in Ávila et al. (2017), can be generalized as follows. For a given set of customers $F^{\mathbb{H}}$, let $v(F^{\mathbb{H}})$ be a lower bound on the minimum number of vehicles needed to service $F^{\mathbb{H}}$. Then, a number of vehicles less than $v(F^{\mathbb{H}})$ cannot service all the customers in $F^{\mathbb{H}}$. Hence, if $v(F^{\mathbb{H}}) \geq 2$, the following inequalities are satisfied by each feasible solution of the DC-CEARP:

$$\sum_{k\in\Omega} z_c^k(F^{\mathbb{H}}) \leq |F^{\mathbb{H}}| - (v(F^{\mathbb{H}}) - |\Omega|), \quad \forall\ \Omega \subseteq \mathbb{K}, \quad 1 \leq |\Omega| \leq v(F^{\mathbb{H}}) - 1, \tag{7.38}$$

$$\sum_{k\in\mathbb{K}\setminus\Omega} x^k(\delta^-(S)) \geq v(F^{\mathbb{H}}) - |\Omega|, \quad \forall\ \Omega \subseteq \mathbb{K}, \quad 0 \leq |\Omega| \leq v(F^{\mathbb{H}}) - 1. \tag{7.39}$$

Note that, for $v(F^{\mathbb{H}}) = 2$, inequalities (7.38) and (7.39) are exactly (7.35) and (7.36)+(7.37) above, respectively. For $v(F^{\mathbb{H}}) = 3$, we have two sets of inequalities (7.38):

$$z_c^k(F^{\mathbb{H}}) \leq |F^{\mathbb{H}}| - 2, \quad \forall k \in \mathbb{K}, \quad \text{and} \tag{7.40}$$

$$z_c^k(F^{\mathbb{H}}) + z_c^{k'}(F^{\mathbb{H}}) \leq |F^{\mathbb{H}}| - 1, \quad \forall k, k' \in \mathbb{K}. \tag{7.41}$$

### 7.3.7 Symmetry breaking inequalities

Let $\{c_1, ..., c_L\}$ be any ordering of the set of customers (for example, according to the distances between them and the depot). The following symmetry breaking inequalities (see Fischetti et al. (1995)) are introduced to avoid equivalent solutions:

$$z_{c_1}^1 = 1 \tag{7.42}$$

$$z_{c_i}^k \leq \sum_{j=1}^{i-1} z_{c_j}^{k-1} \qquad k=3,\ldots,K, \quad i \geq 2 \tag{7.43}$$

$$z_{c_i}^k = 0 \qquad k=i+1,\ldots,K, \quad i=1,\ldots,L-1 \tag{7.44}$$

Inequality (7.42) forces vehicle 1 to service customer $c_1$. Then, vehicle 2 services the first customer in the ordering not serviced by vehicle 1, and so on. Inequalities (7.43) state that if a customer $c_i$ is serviced by vehicle $k$, then at least one 'previous' customer $c_j, j = 1, \ldots, i-1$, has to be serviced by the vehicle $k-1$. Equations (7.44) prevent customers $c_i, i = 1, \ldots, L-1$ from being serviced by vehicles with indices larger than $i$.

## 7.4 The Branch-and-Cut Algorithm

In this section, we present a branch-and-cut algorithm for the DC-CEARP. This new algorithm uses some separation procedures from the methods described in Ávila et al. (2017) and incorporates new ones for some of the inequalities described in Ávila et al. (2017) and for the new inequalities presented in this article. Moreover, an upper bound obtained by the matheuristic algorithm proposed in Corberán et al. (2019) is used.

### 7.4.1 Separation algorithms

In what follows we describe the separation algorithms that have been used to identify the following types of inequalities that are violated by the current LP solution at any iteration of the cutting plane algorithm: connectivity and parity inequalities, disaggregate and $\Omega-$aggregate K-C and K-$C_{02}$, Path-Bridge inequalities, and max-distance constraints. Section 7.4.1 provides a synopsis of the cutting planes and characteristics of their separation procedures.

#### Connectivity inequalities

Several separation procedures have been used to separate connectivity inequalities. The first algorithm, $A1$, separates aggregate connectivity inequalities (7.12). It is based on computing the connected components of the graph induced by the arcs $a$ such that $\sum_{k\in\mathbb{K}} x_a^k \geq \varepsilon$, where $\varepsilon$ is a given parameter. For each weakly connected component, its corresponding aggregate connectivity inequality is checked for violation. We try $\varepsilon = 0, 0.25, 0.5, 0.75$, but a given value is tried only when the previous one did not succeed in finding a violated inequality.

The second heuristic, $A2$, is based on the Gomory-Hu algorithm. It also works on the aggregate graph induced by the sum of the variables corresponding to all the vehicles. If there is a violated (aggregate) connectivity inequality in this graph, it means that there will be a violated (disaggregate) connectivity inequality (7.6) for at least one vehicle.

Two more separation procedures for connectivity inequalities (7.13) have been implemented. The first one, $A3$, works like the first algorithm described in this section, but using the graph induced by the arcs of each single vehicle.

The last algorithm, $A4$, is based on the computation, for each vehicle $k$ and each customer $c$, of the maximum flow on a network containing the arcs for which $x_{ij}^k > 0$ plus an artificial sink and some artificial arcs from the end vertices of the arcs in $H_c$

serviced by vehicle $k$ (i.e. those arcs $a$ such that $y_a^{kc} > 0$) to the sink. The capacity of the arcs is defined as $x_{ij}^k$ for the arcs in the original graph, and as infinity for the artificial ones. The maximum flow from the depot to the sink defines a minimum cutset $(S, V \setminus S)$, with $1 \in V \setminus S$, and the associated connectivity inequality (7.13) is checked for violation.

**Parity inequalities**

We have developed several heuristic algorithms to identify violated parity inequalities. They work as follows.

Given a fractional solution, let $(x^k, y^k, z^k)$ be its part corresponding to vehicle $k$. We build the graph induced by the arcs satisfying $x_a^k - \bar{y}_a^k \geq \varepsilon$, if $a$ is required, and $x_a^k \geq \varepsilon$ otherwise, where $\bar{y}_a^k = \max_{c \in \mathbb{H}} \{y_a^{kc} : a \in H_c\}$, i.e. the maximum value of $y_a^{kc}$ among the customers serviced by arc $a$. Let $S_1, \ldots, S_r$ be the sets of vertices of the weakly connected components of the induced graph. For each cutset $\delta(S_i)$, we now try to select the set of customers $F^{\mathbb{H}} = \{c_1, \ldots, c_q\}$ and the corresponding sets of arcs $\mathcal{F} = \{F_{c_1}, \ldots, F_{c_q}\}$. Two different strategies have been implemented in order to find these sets.

In **strategy 1** (algorithm $A5$), we create a list of pairs of required arcs and customers $(a, c)$ such that $a \in \delta(S_i) \cap H_c$. We order this list according to the value of $y_a^{kc}$ in a decreasing order. Starting from the first pair $(a, c)$ of the list, we iteratively add $c$ to $F^{\mathbb{H}}$ and $a$ to $F_c$ if neither arc $a$ nor customer $c$ have been previously selected. Once the set of customers has been built, we try to enlarge sets $F_c$ by including each unselected arc $a$ of the list in the set $F_c$ with maximum $y_a^{kc}$.

Now, for each vehicle $k$, we calculate $x^k(\delta(S)) - \sum_{i=1}^{q} 2y^{kc_i}(F_{c_i})$. If this value is less than 0, we add $k$ to the set of chosen vehicles $\Omega$.

If $\sum_{k \in \Omega} \sum_{e \in F_c} y_e^{kc} < 0.5$ for some customer $c \in F^{\mathbb{H}}$, this customer is removed from $F^{\mathbb{H}}$. If $|F^{\mathbb{H}}|$ is even, we add or remove one more customer according to the value of $\sum_{k \in \Omega} \sum_{e \in F_c} y_e^{kc}$ in order to make $|F^{\mathbb{H}}|$ odd. For each removed customer $c$, the arcs that belonged to $F_c$ are studied to see if they can be included in another arc subset of $\mathcal{F}$.

Finally, we check if the corresponding $\Omega$-aggregate parity inequality (7.17) is violated.

**Strategy 2** (algorithm $A6$) considers only the cutsets $\delta(S_i)$ for which $x^k(\delta(S_i) \cap A_R)$ is close to an odd number, i.e. $2n + 0.75 \leq x^k(\delta(S_i) \cap A_R) \leq 2n + 1.25$ for some $n \in \{1, 2, \ldots\}$. Let us call $A_{S_i} = \{a \in \delta(S_i) \cap A_R : \sum_{c : a \in H_c} y_a^{kc} > 0\}$ and $\mathbb{H}_{S_i} = \{c \in \mathbb{H} : \sum_{a \in \delta(S_i) \cap A_R} y_a^{kc} > 0\}$, which denote the set of required arcs in cutset $\delta(S_i)$ servicing

some customer and the set of customers that are serviced by some arc in the cutset, respectively.

We calculate $\sum_{a \in A_{S_i}} y_a^{kc}$ for all the customers in $\mathbb{H}_{S_i}$ and select the customer $c$ that maximizes this value. This customer is added to $F^{\mathbb{H}}$ and $F_c = H_c \cap A_{S_i}$. Now we update $A_{S_i}$ by $A_{S_i} \setminus F_c$ and $\mathbb{H}_{S_i}$ by $\mathbb{H}_{S_i} \setminus \{c\}$ and repeat the procedure for choosing the following customers until $|F^{\mathbb{H}}| = 2n + 1$.

As in strategy 1, we include in $\Omega$ all the vehicles $k$ for which $x^k(\delta(S)) - \sum_{i=1}^{q} 2y^{kc_i}(F_{c_i}) < 0$ and check if the corresponding $\Omega$-aggregate parity inequality (7.17) is violated.

In order to separate parity inequalities (7.16) when $|\Omega| = K$, algorithm $A7$ uses a similar procedure with strategy 2 adapted to the graph induced by the arcs satisfying $\sum_{k \in \mathbb{K}} x_a^k - 1 \geq \varepsilon$, if $a$ is required, and $\sum_{k \in \mathbb{K}} x_a^k \geq \varepsilon$ otherwise.

We have also tried an alternative method for selecting the set of customers $F^{\mathbb{H}}$ based on the solution of the following integer program. As before, we define $\mathbb{H}_{S_i}$ as the set of customers that are serviced by some arc in the cutset $\delta(S_i)$. For each customer $c \in \mathbb{H}_{S_i}$, we define a binary variable $\mu_c$ that takes value 1 if $c$ is included in $F^{\mathbb{H}}$ and 0 otherwise. Let us define $w_c = \sum_{a \in \delta(S_i)} y_a^{kc}$ for each customer $c \in \mathbb{H}_{S_i}$ and consider two customers $c_r, c_s$ as incompatible if there is an arc $a \in \delta_R(S_i)$ such that $y_a^{kc_r} > 0$ and $y_a^{kc_s} > 0$. Then, we solve the following IP:

$$\text{Maximize} \qquad \sum_{c \in \mathbb{H}_{S_i}} w_c \, \mu_c$$

s.t.:

$$\sum_{c \in \mathbb{H}_{S_i}} \mu_c \equiv odd \tag{7.45}$$

$$\mu_{c_r} + \mu_{c_s} \leq 1 \qquad \forall \, c_r, c_s \text{ incompatible} \tag{7.46}$$

$$\mu_c \in \{0, 1\} \qquad \forall c \in \mathbb{H}_{S_i} \tag{7.47}$$

In order to study the performance of the above method, we have compared it with the heuristic procedure for selecting $F^{\mathbb{H}}$. On a sample of 27 randomly selected instances, the IP-based method used, on average, 123.53 seconds per instance to find, on average, 0.22 violated parity inequalities per call, while the heuristic method found, on average, 0.13 parity cuts in 0.46 seconds of computing time. Based on these results, we have decided not to use the IP-based method.

### K-C, K-C$_{02}$ and Path-Bridge inequalities

Algorithm $A8$ looks first for the graph structure associated with the disaggregate K-C inequalities (7.20) and (7.21). Again, let $(x^k, y^k, z^k)$ be the part corresponding to vehicle $k$ of a given fractional solution. Let $G^k$ be the graph induced by the arcs $a$ with $x_a^k > 0$ and label the depot and the arcs $a \in A_R$ such that $x_a^k \geq \varepsilon$ and $y_a^k \geq x_a^k/2$ as 'required', where $\varepsilon$ is a given parameter. We compute the connected components induced by these arcs and the depot. Let us call $C_i$ to these components. We then apply a procedure based on that described in Corberán et al. (2001) for the undirected GRP to obtain the sets $M_0, M_1, M_2, \ldots, M_Q$, which consists of, given a component $C_i$, checking if it is connected to two different components by arcs with $x_a^k > 0$. For such a component, we try to split it in two parts such that each part is connected to a different component. These two parts will be the "seeds" for defining sets $M_0$ and $M_Q$. Now we shrink these seeds and the remaining components into a single vertex each and compute a spanning tree by iteratively adding the arc of maximum weight not forming a cycle (and not connecting the seeds). This tree is transformed into a path linking the seeds by (iteratively) shrinking each non-seed vertex with degree one into its (unique) adjacent vertex. If the length of the path is at least 3, the vertices of the path define the seeds for sets $M_0, M_1, \ldots, M_Q$. All the vertices of $G$ that do not belong to a set $M_i$ yet are iteratively assigned to a set $M_i$ to which they are adjacent.

Set $\mathcal{F}$ is formed by some sets of 'required' arcs in $M_0 \cup M_Q$ associated with different customers. For each set $M_j$, $j = 1, \ldots, Q-1$, non containing the depot, we define $G_j$ as the set of arcs $G_j = H_{c_j} \cap (A(M_j) \cup \delta(M_j))$. If the corresponding K-C inequality for vehicle $k$ is not violated, we try to improve the inequality by shrinking some consecutive sets $M_j$. Several values for $\varepsilon$ have been tried and, after some computational testing, we finally decided to set $\varepsilon = 0.2$.

At this point, a K-C structure has been found, and its corresponding disaggregate K-C inequality for vehicle $k$ can, or not, be violated. Since inequality (7.20) (if the depot belongs to $M_0 \cup M_Q$, otherwise it would be similar) can be written as (7.22), we evaluate its left hand side for each vehicle $k' = 1, \ldots, K$. Then we include in $\Omega$ all the vehicles $k'$ for which this expression is less than 0 and check if the resulting $\Omega$-aggregate K-C inequality (7.23) or (7.24) is violated.

Any K-C structure found by algorithm $A8$ is used to look for violated K-C$_{02}$ inequalities (algorithm $A9$). As before, the inequality for each single vehicle $k'$ is checked and those vehicles for which the left hand side is negative are included into $\Omega$. The separation of 2 Path-Bridge inequalities is done with a similar procedure, algorithm $A10$, that is not described here for the sake of brevity.

### Max-distance inequalities

Two heuristic algorithms are used to separate max-distance inequalities. The first heuristic, $A11$, is the one described in Ávila et al. (2017) for separating inequalities (7.36). If a violated max-distance constraint (7.36) is found, at least one of the inequalities (7.37) is also violated and it is added. Furthermore, the corresponding inequality (7.38) is also added.

The second heuristic, $A12$, looks for violated inequalities (7.38). It is designed to cut fractional solutions in which, for a vehicle $k$, several $z_c^k$ variables take values close to 1 and another one takes a value close to 0.5. It works as follows.

Given a fractional solution associated with vehicle $k$, $(x^k, y^k, z^k)$, let $\{c_1, c_2, \ldots, c_q\}$ be the set of customers such that $z_{c_1}^k \geq z_{c_2}^k \geq \ldots \geq z_{c_q}^k \geq 0.5$. We define $F^{\mathbb{H}} = \{c_1, c_2, \ldots, c_f\}$, where $f$ is the maximal number such that $z_c^k(F^{\mathbb{H}}) > |F^{\mathbb{H}}| - 1 + \varepsilon$ (initially we set $\varepsilon = 0.5$), and we call 'potential customers' to the remaining $\{c_{f+1}, c_{f+2}, \ldots, c_q\}$. We check if $v(F^{\mathbb{H}})$ is greater than one and, therefore, the corresponding inequality (7.38) is violated. Otherwise, for each potential customer $\overline{c} \in \{c_{f+1}, c_{f+2}, \ldots, c_q\}$, we iteratively consider the set $\overline{F}^{\mathbb{H}} = F^{\mathbb{H}} \cup \{\overline{c}\}$ and check if $v(\overline{F}^{\mathbb{H}})$ is greater than one. Finally, if no violated inequality has been found for any set $\overline{F}^{\mathbb{H}}$, we set $\varepsilon = 0$ and define the set $F^{\mathbb{H}}$ as above ($f$ is now the maximal number such that $z_c^k(F^{\mathbb{H}}) > |F^{\mathbb{H}}| - 1$) and check if $v(F^{\mathbb{H}})$ is greater than one. If a subset $F^{\mathbb{H}}$ (or $\overline{F}^{\mathbb{H}}$) for which the corresponding inequality (7.38) is violated is found, this inequality is added. Then, we look for the cutset of minimum weight between the depot and the arcs of the customer in $F_H$ and the corresponding max-distance inequalities (7.36) and (7.37) are checked for violation.

Given a set of customers $F^{\mathbb{H}}$, the value of $v(F^{\mathbb{H}})$ (either the number of vehicles needed to service $F^{\mathbb{H}}$ or a lower bound) is computed by solving the corresponding CEARP using the branch-and-cut algorithm in Ávila et al. (2016b). Since solving the CEARP instances to optimality can be time consuming, we have limited the execution time of the CEARP solver to 10 seconds.

### Inequalities and separation procedures: a summary

Table 7.1 summarizes the separation procedures described before and provides information on their computational complexity and the references where they were proposed or used. In particular, the first column shows the inequalities class, while column two gives the exact family of inequalities being separated. Column "Separ.

Proc." provides the name of the separation procedure used and column "Type" indicates if the procedure is heuristic ("H") or almost exact ("AE"). This last type means that the algorithm is an adaptation of an exact procedure for identifying a violated inequality with similar characteristics. For example, algorithm $A4$ is based on the exact separation of inequalities

$$x^k(\delta^+(S)) \geq \sum_{(i,j)\in H_c} y_{ij}^{kc}, \quad \forall S \subset V\setminus\{1\},\ \forall c \in \mathbb{H},\ \forall k \in \mathbb{K}$$

However, note that the sum in this inequality is done for all the arcs $(i, j) \in H_c$, while in inequalities (7.13) the sum is for all $(i, j) \in H_c \setminus A(V \setminus S)$. The computational complexity of the algorithms is given in Column 5. An asterisk (*) means that the reported value is the computational complexity of the corresponding separation algorithm if no call to the B&C algorithm described in Ávila et al. (2016b) is done. If the B&C is executed to compute the minimum number of vehicles needed to service a given subset of customers, the resulting computational effort is non-polynomial (but each call is limited to 10 seconds). The last two columns indicate if the procedure has already been used in other works and report the corresponding references. A "No/Yes" entry indicates that some new parts have been added in this contribution to the existing procedures and "[TP]" is used to refer to this contribution.

| Class | Inequalities | Separ. Proc. | Type | Complexity | New? | Reference |
|---|---|---|---|---|---|---|
| Connectivity | (7.12) | A1 | H | $O(\|A\|\|\mathbb{H}\|)$ | No | Corberán et al. (2001) |
| | (7.6) | A2 | AE | $O(\|V\|^3\|A\|)$ | No | Ávila et al. (2017) |
| | (7.13) | A3 | H | $O(K\|A\|\|\mathbb{H}\|)$ | No/Yes | Ávila et al. (2017), [TP] |
| | (7.13) | A4 | AE | $O(K\|\mathbb{H}\|\|V\|^2\|A\|)$ | No/Yes | Ávila et al. (2017), [TP] |
| Parity | (7.17) | A5 | H | $O(K\|\mathbb{H}\|\|V\|\|A\|)$ | Yes | [TP] |
| | (7.17) | A6 | H | $O(K\|\mathbb{H}\|\|V\|\|A\|)$ | Yes | [TP] |
| | (7.16) | A7 | H | $O(\|V\|\|A\|\|\mathbb{H}\|)$ | Yes | [TP] |
| K-C | (7.20), (7.21), (7.23), (7.24) | A8 | H | $O(K\|V\|^2\|A\|)$ | No/Yes | Corberán et al. (2001), [TP] |
| K-$C_{02}$ | (7.25), (7.26), (7.27), (7.28) | A9 | H | $O(K\|V\|^2\|A\|)$ | No/Yes | Corberán et al. (2003), [TP] |
| 2 Path-Bridge | (7.29), (7.30), (7.33), (7.34) | A10 | H | $O(K\|V\|^2\|A\|)$ | No/Yes | Corberán et al. (2001), [TP] |
| Max-distance | (7.36), (7.37), (7.38) | A11 | H | $O(\|\mathbb{H}\|)*$ | No | Ávila et al. (2017) |
| | (7.36), (7.37), (7.38) | A12 | H | $O(K)*$ | Yes | [TP] |

Table 7.1: Inequalities and separation procedures

### 7.4.2 Comparison of separation strategies and cutting-plane algorithms

To analyze the contribution of the valid inequalities and the separation algorithms presented in the previous sections, we compare the gaps in the root node and the performance profiles (Dolan and More (2002)) of the different versions of our branch-and-cut procedure using different combinations of separation algorithms.

Let $\mathcal{S}$ be the set of versions of our algorithm and $\mathcal{P}$ the set of instances selected for this comparison. Then, for each version $s \in \mathcal{S}$, we calculate $GAP0_s = \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}(BKS_p - LB0_{p,s})/LB0_{p,s} * 100$, where $BKS_p$ denotes the value of the optimal or best known solution obtained by any version for instance $p$ and $LB0_{p,s}$ the lower bound obtained by $s$ at the root node. We also compute the *performance ratio* $r_{p,s} = t_{p,s}/\min\{t_{p,s} : s \in \mathcal{S}\}$, where $t_{p,s}$ is the computing time required by algorithm $s$ to solve instance $p$. If algorithm $s$ is not able to solve the instance $p$ within the time limit, we set $r_{p,s} = \infty$. Thus, the *performance profile* of each version $s$,

$$\rho_s(\tau) = \frac{|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}|}{|\mathcal{P}|},$$

describes the percentage of instances that can be solved by $s$ within a factor $\tau \geq 1$ compared to the fastest algorithm. Note, for example, that $\rho_s(1)$ is the percentage of instances for which algorithm $s$ is the fastest and that $\rho_s(\infty)$ is the percentage of instances that are solved by algorithm $s$ within the time limit.

We started with a "full version" of the branch and cut, denoted by $V1234$, with the following characteristics. The initial LP relaxation contains all the inequalities in the formulation, except for the connectivity inequalities (7.6) that are exponential in number and, hence, only the following subset of them are included:

$$x^k(\delta^-(S_c)) \geq z_c^k, \quad \forall k \in \mathbb{K},\ \forall c \in \mathbb{H},$$

where $S_c$ is the set of vertices incident with the arcs in $H_c$.

Furthermore, the symmetry breaking inequalities (7.42)-(7.44), some max-distance inequalities (7.38) and (7.39) associated with some subsets of customers that cannot be serviced with a single vehicle, and inequalities $x^k(\delta^+(1)) \geq 1,\ \forall k \in \mathbb{K}$, which force each vehicle to leave the depot, are included. At each iteration of the cutting-plane algorithm in the root node, the separation procedures described above are applied using the following general scheme and the violated inequalities found are added to the LP relaxation:

1 All the heuristic separation algorithms for connectivity inequalities (A1-A4) are applied. The algorithm based on flow computations (A4) is used only if the other ones fail to find violated inequalities.

2 Heuristic parity separation algorithms (A5-A7).

3 Algorithms for separating K-C, K-$C_{02}$, and Path-Bridge inequalities (A8-A10) are applied for each vehicle $k$ only if no violated connectivity inequalities have been found for this vehicle.

4 Heuristic algorithms A11 and A12 for separating max-distance inequalities.

Only the fastest separation algorithm for disaggregate connectivity inequalities (A3) is applied in the nodes of the branch-and-cut tree.

The above cutting-plane procedure is applied until no new violated inequalities are found. When this happens, we branch using the Strong Branching strategy implemented in CPLEX with higher priority given to the $z_c^k$ and $y_{ij}^{kc}$ variables.

All other B&C versions are based on different cutting-plane algorithms associated with different separation strategies. The B&C algorithms were tested on a subset of 48 instances taken from the four sets of DC-CEARP instances proposed in Ávila et al. (2017) and whose characteristics are described in Section 7.5.1. Twelve instances, three for each value of $k \in \{2, 3, 4, 5\}$, were chosen at random from each subset. The experiments were performed on a desktop PC with an Intel(R) Core(TM) i7 at 3.4GHz CPU with 32GB RAM running Windows 10 Enterprise 64 bits using a single thread. The algorithms were coded in C++ combined with CPLEX 12.10 and all the experiments were carried out with a time limit of 7200 seconds.

We first studied the impact of connectivity inequalities and their separation procedures. To do this, we compared the V1234 B&C with three new versions. In the first version, called V234, we remove all the separation algorithms for connectivity inequalities (A1-A4), except for algorithm A1 when the obtained solution is integer. The second version, V1(a)+234, uses all the separation algorithms except for A2, while the last one, V1(b)+234, uses all the separation algorithms except for A4. Note that A2 and A4 are the most time-consuming procedures.

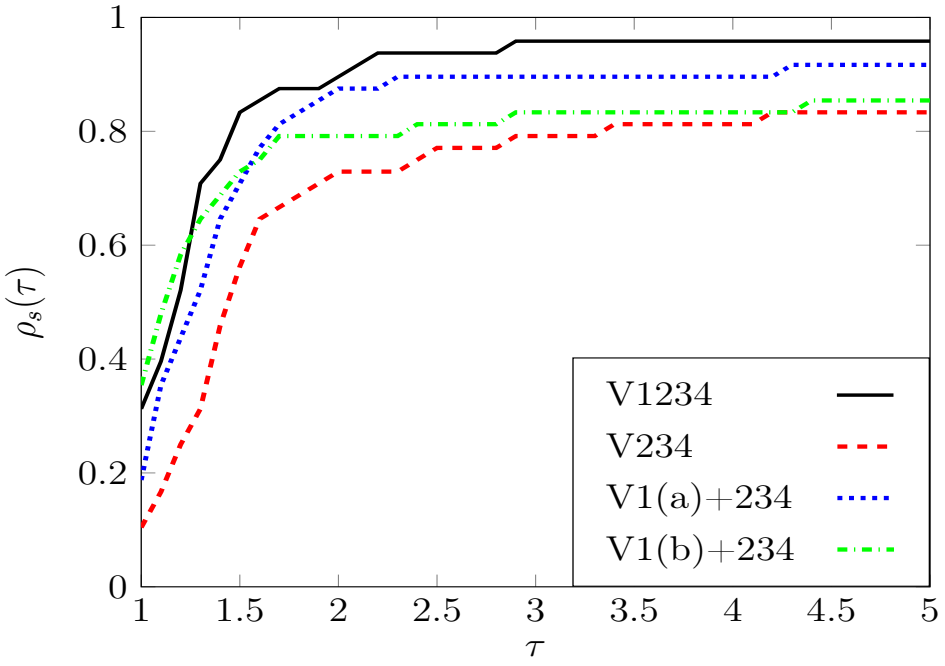



Figure 7.7: Impact of the connectivity inequalities: Performance profile

Figure 7.7 shows the performance profile of the four compared versions and Table 7.2 reports for each version the number of optima obtained (out of 48 instances), the average gap in the root node, and the average computing time spent at the root node and the average total computing time in seconds. V1234, as expected, and surprisingly V1(a)+234, are the best versions in terms of gap, although this last version shows a worse performance profile and worse behavior in terms of averages and number of optima. The other two versions, V234 and V1(b)+234, are clearly dominated by $V1234$. Therefore, we decided to include all separation procedures for connectivity in the final version of the B&C.

| | # opt | Gap0 (%) | Time0 (scs) | Time (scs) |
|---|---|---|---|---|
| $V1234$ | 46 | 5.874 | 252.38 | 1031.19 |
| $V234$ | 41 | 11.560 | 198.83 | 1654.31 |
| $V1(a) + 234$ | 44 | 5.824 | 265.86 | 1209.48 |
| $V1(b) + 234$ | 41 | 8.693 | 192.62 | 1491.66 |

Table 7.2: Results on the subset of 48 instances - connectivity

Then, we studied the effect of parity inequalities and their separation by comparing the full version V1234 with two versions obtained from it by removing all the separation algorithms for parity inequalities (A5-A7), version V134, and removing algorithms A5 and A7 (V1+2(a)+34). The results obtained are summarized in Figure 7.8 and Table 7.3.

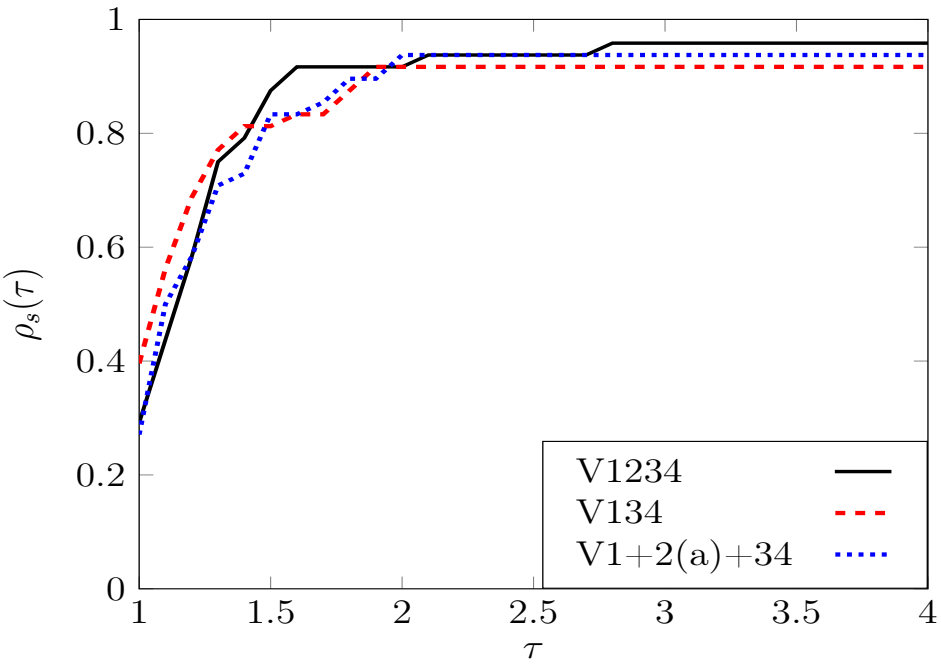



Figure 7.8: Impact of the parity inequalities: Performance profile.

| | # opt | Gap0 (%) | Time0 (scs) | Time (scs) |
|---|---|---|---|---|
| $V1234$ | 46 | 5.874 | 252.38 | 1031.19 |
| $V134$ | 44 | 6.065 | 258.00 | 1069.87 |
| $V1+2(a)+34$ | 45 | 6.034 | 268.13 | 1165.09 |

Table 7.3: Results on the subset of 48 instances - parity

As Figure 7.8 shows, all three versions compared have similar performance profiles. However, the gaps in the root node and other measures reported in Table 7.3 for V134 and V1+2(a)+34 are worse than those of the full version, and therefore none of the latter versions is considered interesting.

We also considered different options regarding the max-distance inequalities. Here, three new versions were implemented. In the first one, V123+4(a), we removed the separation algorithm A11, while algorithm A12 was removed in the second one V123+4(b). The third version, V123, did not include any separation algorithm for max-distance inequalities. These three versions are compared again with version V1234. Performance profiles, average gaps in the root node, and other measures are presented in Figure 7.9 and Table 7.4.

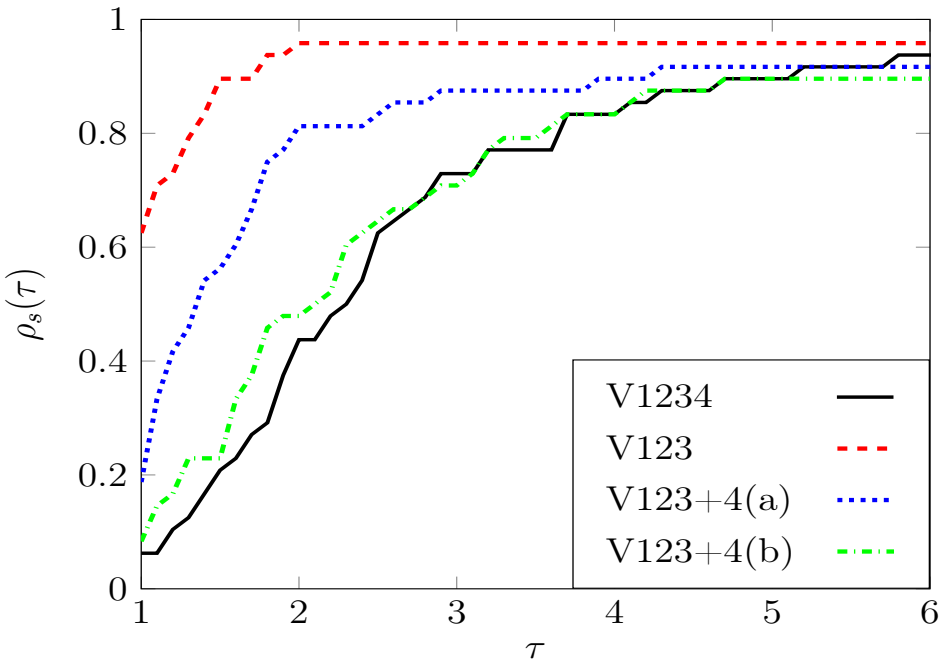



Figure 7.9: Impact of the max-distance inequalities: Performance profile.

| | # opt | Gap0 (%) | Time0 (scs) | Time (scs) |
|---|---|---|---|---|
| $V1234$ | 46 | 5.874 | 252.38 | 1031.19 |
| $V123$ | 46 | 9.835 | 114.34 | 902.39 |
| $V123+4(a)$ | 46 | 7.738 | 136.99 | 1036.11 |
| $V123+4(b)$ | 46 | 6.655 | 241.18 | 1054.21 |

Table 7.4: Results on the subset of 48 instances - max-distance

From Figure 7.9 we can see that V123 is the fastest version in 60% of the instances, followed by V123+4(a), although they are the two versions with the worst gaps. V123+4(b) is not an interesting option because its performance profile is similar to that of V1234 but it shows worse gaps. The V123+4(a) version has a better performance profile than the full version but a worse average gap, and it is clearly dominated in terms of computing time and performance profile by V123. Therefore, we selected V123 as the most interesting option among the three tested versions.

Finally, we compared a new version V124 resulting from removing the separation algorithms A8-A10 for K-C, K-$C_{02}$, and Path-Bridge inequalities, with the most promising versions obtained from the previous experiments, V1234 and V123. Note that separation algorithms A8, A9, and A10 have many parts in common, and thus removing only some of them would produce no benefit in terms of the overall algorithm efficiency.

Figure 7.10 shows the performance profiles for the three versions. Version V123 is the fastest one to reach the optimal solution in more than 80% of the instances and can

optimally solve all 46 instances in less that 2 times the time of the fastest version. The performance profile of the other two versions is similar and it can be seen that they reach the 46 optimal solutions only with factors $\tau = 20$ and $\tau = 21$. As for the average gaps in the root node, V124 does not improve the results obtained by V1234 either. Looking at the Table 7.5, we can see that V124 has no advantage over V1234 and is therefore not considered an interesting alternative. Overall, version V123, although produces greater gaps at the root node, is considerably faster and has a better performance profile than V1234. Therefore, we decided to use version V123 for our final computational experiments.

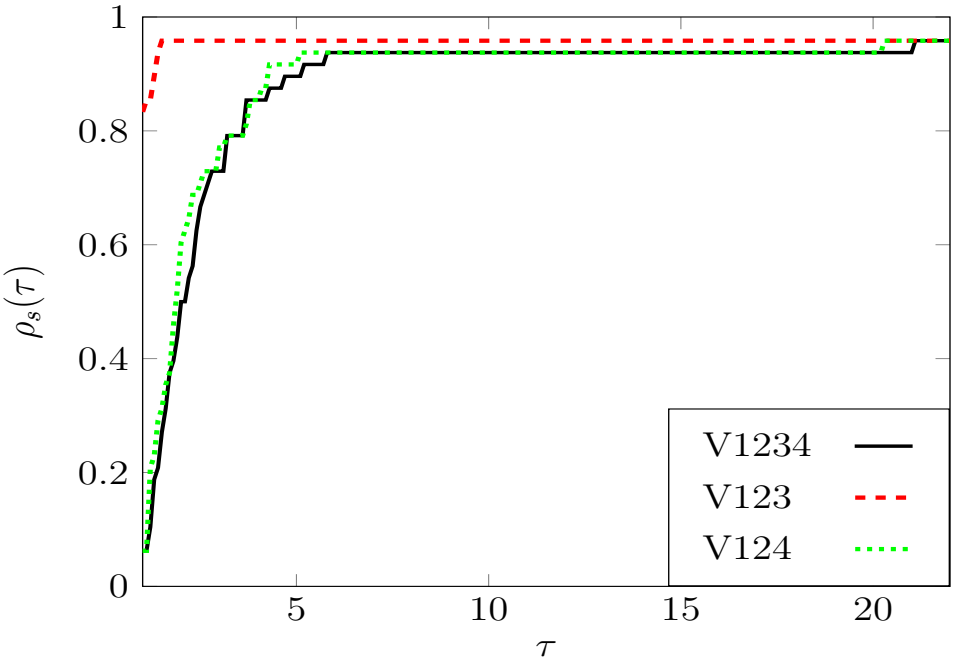



Figure 7.10: Impact of the K-C, K-$C_{02}$ and Path-Bridge inequalities: Performance profile.

| | # opt | Gap0 (%) | Time0 (scs) | Time (scs) |
|---|---|---|---|---|
| $V1234$ | 46 | 5.874 | 252.38 | 1031.19 |
| $V123$ | 46 | 9.835 | 114.34 | 902.39 |
| $V124$ | 46 | 6.064 | 228.98 | 1127.72 |

Table 7.5: Results on the subset of 48 instances - K-C, K-$C_{02}$ and Path-Bridge

## 7.5 Computational experiments

In this section we study the performance of the final version (V123) of the branch-and-cut algorithm, *Algorithm 1* in what follows. As in the previous analysis, the experiments were performed on a desktop PC with an Intel(R) Core(TM) i7 at 3.4GHz CPU

with 32GB RAM running Windows 10 Enterprise 64 bits. Again, we used CPLEX 12.10 with a single thread. The performance of *Algorithm 1* has been compared with that of the best of four branch-and-cut procedures described in Ávila et al. (2017), *Algorithm 2* in what follows, and, for illustrative purposes, with the full version V1234, denoted as *Algorithm 0*. *Algorithm 2* has been executed on the same machine and using the same version of CPLEX.

All the experiments were carried out with a time limit of two hours. CPLEX heuristic algorithms were turned off, and CPLEX own cuts, including zero-half cuts, were activated in automatic mode. The optimality gap tolerance was set to zero, best bound strategy was selected and CPLEX presolve phase was reapplied at the end of the root node. The instances used and the computational results obtained are described in what follows.

### 7.5.1 Instances

We have tested our branch-and-cut algorithm on the four sets of DC-CEARP instances proposed in Ávila et al. (2017). The graphs of the two first sets of instances, *Random50* and *Random75*, were generated randomly and have 50 and 75 vertices respectively. Sets *Albaida* and *Madrigueras* are based on the street networks of these two Spanish towns. As pointed out in Ávila et al. (2017), generating the value of $D_{max}$ for each instance is a hard task, because depending on this value, the instance can be infeasible or trivial (some of the vehicles are not needed). A detailed description of how these values have been generated can be found in that paper. The characteristics of these 251 instances are summarized in Table 7.6. The complete data, including the values of $D_{max}$ and the number of vehicles with the corresponding best solutions found, can be downloaded from *http://www.uv.es/corberan/instancias.htm* in the class "Distance-Constrained CEARP".

| | | $\|A\|$ | | $\|A_R\|$ | | $\|A_{NR}\|$ | | $\|\mathbb{H}\|$ | |
|---|---|---|---|---|---|---|---|---|---|
| | $\|V\|$ | Min | Max | Min | Max | Min | Max | Min | Max |
| *Random50* | 50 | 296 | 300 | 105 | 292 | 7 | 193 | 10 | 97 |
| *Random75* | 75 | 448 | 450 | 143 | 438 | 10 | 305 | 15 | 140 |
| *Albaida* | 116 | 259 | 305 | 124 | 172 | 109 | 162 | 18 | 33 |
| *Madrigueras* | 196 | 453 | 544 | 224 | 305 | 197 | 281 | 22 | 47 |

Table 7.6: Characteristics of the instances

### 7.5.2 Computational Results

The computational results obtained with *Algorithm 0*, *Algorithm 1*, and *Algorithm 2*, are shown in Table 7.7, where instances have been grouped by number of vehicles and number of customers, which are shown in columns 1 and 2. Column 3 reports the number of instances of each subset. For all algorithms, the columns labeled '# opt', 'Gap0 (%)', and 'Time' report the number of optimal solutions found, the average gap in the root node, and the average computing time in seconds, respectively. The bold rows at each group of instances with the same number of vehicles show the total number of the 'inst' and '# opt' columns, and the average values for the remaining columns. The last row of the table summarizes the results for all the instances.

As can be seen in Table 7.7, the number of optima obtained with *Algorithm 1* is very good (234 out of 251 instances) and is a bit better than those obtained with *Algorithm 0* (231 optima) and *Algorithm 2* (229 optima), although the average gap in the root node is slightly higher than that of *Algorithm 2* (8.94% versus 8.42%), and, as expected, higher than the one obtained using all the separation procedures described in Section 7.4.1 (6.5%). However, it is in the computing times where we can appreciate the main differences. The average computing time is 976.3 seconds with *Algorithm 1* versus 1080.1 seconds obtained with *Algorithm 2* and 1300.9 of *Algorithm 0*. Although for 2 and 3 vehicles the times for *Algorithm 2* are better on average, when the number of vehicles increases, the times of *Algorithm 1* are significantly lower. Tables 7.10, 7.11, 7.12, and 7.13 report the same computational results disaggregated for the sets Random50, Random75, Albaida, and Madrigueras, respectively.

To study the running times in more detail, we compare the performance profiles of the three branch-and-cut algorithms (see Figure 7.11). Comparing the performance ratios of the three methods at $\tau = 1$, we observe that *Algorithm 1* is the fastest one in almost 80% of the instances, while *Algorithm 0* is the slowest one. As $\tau$ increases, the difference between *Algorithm 1* and *Algorithm 2* decreases, but note that five more instances can be solved using *Algorithm 1*.

| | | | *Algorithm 2* (Ávila et al. (2017)) | | | *Algorithm 1* | | | *Algorithm 0* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Veh | $\|\mathbb{H}\|$ | inst | # opt | Gap0(%) | Time | # opt | Gap0(%) | Time | # opt | Gap0(%) | Time |
| | [10,21] | 18 | 18 | 1.80 | 19.2 | 18 | 2.4 | 8.8 | 18 | 0.74 | 36.5 |
| | [12,30] | 21 | 21 | 1.62 | 111.6 | 21 | 2.6 | 75.2 | 21 | 1.19 | 240.5 |
| 2 | [31,46] | 16 | 16 | 1.19 | 109.3 | 16 | 1.9 | 86.9 | 16 | 1.12 | 256.5 |
| | [47,140] | 17 | 17 | 1.64 | 365.1 | 16 | 1.6 | 712.5 | 16 | 1.74 | 1030.8 |
| | | **72** | **72** | **1.57** | **147.8** | **71** | **2.14** | **211.7** | **71** | **1.19** | **379.7** |
| | [10,21] | 17 | 17 | 7.75 | 40.4 | 17 | 8.5 | 21.0 | 17 | 4.67 | 55.5 |
| | [12,30] | 21 | 21 | 7.07 | 238.7 | 21 | 7.7 | 140.4 | 21 | 5.70 | 324.9 |
| 3 | [31,46] | 16 | 16 | 4.74 | 696.8 | 16 | 5.3 | 578.7 | 16 | 4.32 | 775.0 |
| | [47,140] | 17 | 16 | 5.23 | 1218.9 | 16 | 4.6 | 1477.1 | 16 | 5.17 | 1718.2 |
| | | **71** | **70** | **6.27** | **529.2** | **70** | **6.61** | **530.7** | **70** | **5.02** | **695.5** |
| | [10,21] | 12 | 12 | 14.53 | 49.6 | 12 | 15.8 | 27.3 | 12 | 6.88 | 71.9 |
| | [12,30] | 19 | 19 | 14.33 | 766.9 | 19 | 15.6 | 344.2 | 19 | 11.20 | 1236.5 |
| 4 | [31,46] | 16 | 13 | 11.65 | 2253.8 | 16 | 11.0 | 908.1 | 14 | 10.05 | 1790.2 |
| | [47,140] | 17 | 11 | 9.45 | 3178.5 | 13 | 8.6 | 3454.6 | 13 | 9.02 | 3862.8 |
| | | **64** | **55** | **12.40** | **1644.7** | **60** | **12.66** | **1251.9** | **58** | **9.52** | **1854.2** |
| | [10,21] | 9 | 9 | 19.35 | 73.9 | 9 | 24.0 | 60.6 | 9 | 6.39 | 100.7 |
| | [12,30] | 10 | 10 | 19.63 | 956.8 | 10 | 22.5 | 589.0 | 10 | 17.04 | 983.1 |
| 5 | [31,46] | 10 | 5 | 15.64 | 3963.6 | 8 | 15.2 | 2858.8 | 7 | 14.01 | 3684.5 |
| | [47,140] | 15 | 8 | 15.61 | 4516.9 | 6 | 14.5 | 5132.1 | 6 | 14.13 | 5571.8 |
| | | **44** | **32** | **17.29** | **2673.3** | **33** | **18.45** | **2545.6** | **32** | **13.18** | **2980.9** |
| TOTAL | | **251** | **229** | **8.42** | **1080.1** | **234** | **8.94** | **976.3** | **231** | **6.50** | **1300.9** |

Table 7.7: Computational results for all the instances grouped by number of vehicles and customers

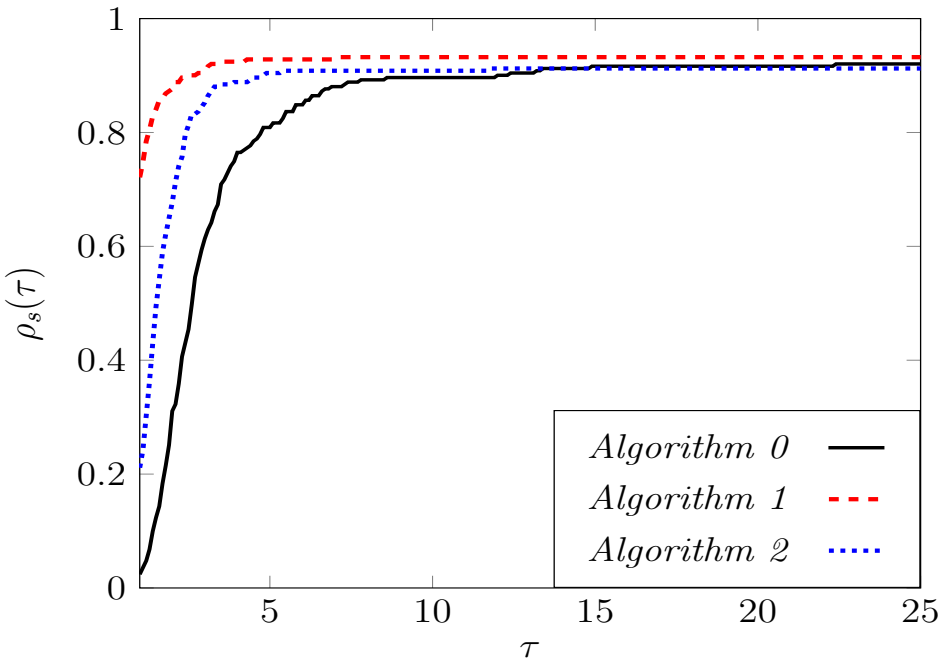



Figure 7.11: Performance profile

Table 7.7 does not report the average final gaps since most instances are optimally solved by all algorithms. Instead, Table 7.8 compares the average gaps in the root node and the average final gaps obtained by the three procedures in the 11 instances that are not solved by any of the algorithms, while Table 7.9 provides the same information

but in the 27 instances that have not been optimally solved by at least one of the algorithms. Note that in these harder instances, *Algorithm 1* obtains better gaps at the root node, as well as final gaps. Table 7.9 also reports the number of instances solved optimally by each algorithm.

| | Gap0(%) | Final Gap(%) |
|---|---|---|
| *Algorithm 0* | 14.20 | 8.98 |
| *Algorithm 1* | 14.36 | 7.80 |
| *Algorithm 2* | 15.35 | 9.23 |

Table 7.8: Results on the 11 instances not solved by any algorithm

| | # opt | Gap0(%) | Final Gap(%) |
|---|---|---|---|
| *Algorithm 0* | 7 | 12.07 | 4.98 |
| *Algorithm 1* | 10 | 11.86 | 3.68 |
| *Algorithm 2* | 5 | 12.94 | 5.27 |

Table 7.9: Results on the 27 instances not solved by at least one algorithm

Looking at the computational results disaggregated by sets of instances, Tables 7.10 to 7.13, we note that the performance of *Algorithm 1* is better than that of *Algorithm 2* in those instances that are based on real street networks like the Albaida and Madrigueras sets. Another particularity of these two sets of instances is that their number of customers is not too large, from 19 to 34 and from 23 to 48 in the Albaida and Madrigueras instances, respectively, because they are defined following "geographical" criteria, as it was assumed that it can occur in real-life problems. *Algorithm 2*, on the other hand, performs better on the Random 50 and 75 sets, which were randomly generated and have a larger number of customers (many of them defined by larger subsets of arcs).

| | | | *Algorithm 2* (Ávila et al. (2017)) | | | *Algorithm 1* | | | *Algorithm 0* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Veh | $\|\mathbb{H}\|$ | inst | # opt | Gap0(%) | Time | # opt | Gap0(%) | Time | # opt | Gap0(%) | Time |
| 2 | 10 | 3 | 3 | 1.28 | 4.1 | 3 | 1.65 | 2.4 | 3 | 0.00 | 4.0 |
| | [24,25] | 3 | 3 | 0.48 | 12.9 | 3 | 0.06 | 8.8 | 3 | 0.20 | 23.3 |
| | [45,50] | 3 | 3 | 0.64 | 42.4 | 3 | 1.72 | 33.8 | 3 | 0.87 | 90.8 |
| | [92,97] | 3 | 3 | 2.31 | 124.0 | 3 | 2.24 | 282.2 | 3 | 2.38 | 333.5 |
| | | **12** | **12** | **1.18** | **45.9** | **12** | **1.42** | **81.8** | **12** | **0.86** | **112.9** |
| 3 | 10 | 2 | 2 | 4.89 | 4.6 | 2 | 4.95 | 3.5 | 2 | 0.83 | 5.4 |
| | [24,25] | 3 | 3 | 2.94 | 17.1 | 3 | 7.10 | 14.3 | 3 | 2.57 | 25.2 |
| | [45,50] | 3 | 3 | 4.13 | 41.8 | 3 | 4.17 | 47.3 | 3 | 3.96 | 102.0 |
| | [92,97] | 3 | 3 | 7.36 | 204.2 | 3 | 6.63 | 546.3 | 3 | 7.45 | 665.9 |
| | | **11** | **11** | **4.83** | **72.5** | **11** | **5.78** | **166.4** | **11** | **3.96** | **217.3** |
| 4 | [24,25] | 3 | 3 | 10.13 | 21.0 | 3 | 11.28 | 19.1 | 3 | 8.87 | 43.1 |
| | [45,50] | 3 | 3 | 11.70 | 61.3 | 3 | 10.74 | 91.3 | 3 | 9.59 | 225.5 |
| | [92,97] | 3 | 3 | 9.46 | 296.3 | 3 | 8.82 | 897.8 | 3 | 9.40 | 1286.2 |
| | | **9** | **9** | **10.43** | **126.2** | **9** | **10.28** | **336.0** | **9** | **9.29** | **518.3** |
| 5 | [45,50] | 1 | 1 | 23.56 | 177.3 | 1 | 23.71 | 362.5 | 1 | 16.78 | 247.6 |
| | [92,97] | 2 | 2 | 15.06 | 1257.2 | 1 | 14.39 | 3838.7 | 2 | 12.21 | 3652.9 |
| | | **3** | **3** | **17.89** | **897.2** | **2** | **17.49** | **2680.0** | **3** | **13.74** | **2517.9** |
| TOTAL | | **35** | **35** | **6.14** | **147.9** | **34** | **6.45** | **396.4** | **35** | **5.11** | **456.1** |

Table 7.10: Results for the Random50 instances

| | | | *Algorithm 2* (Ávila et al. (2017)) | | | *Algorithm 1* | | | *Algorithm 0* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Veh | $\|\mathbb{H}\|$ | inst | # opt | Gap0(%) | Time | # opt | Gap0(%) | Time | # opt | Gap0(%) | Time |
| 2 | 15 | 3 | 3 | 0.00 | 8.6 | 3 | 0.43 | 5.9 | 3 | 0.00 | 15.2 |
| | [36,37] | 3 | 3 | 1.34 | 38.5 | 3 | 1.66 | 25.7 | 3 | 1.02 | 74.4 |
| | [70,75] | 3 | 3 | 1.75 | 117.1 | 3 | 1.40 | 165.3 | 3 | 1.79 | 568.2 |
| | [138,140] | 3 | 3 | 2.42 | 1301.6 | 2 | 1.93 | 3224.6 | 2 | 2.48 | 3727.9 |
| | | **12** | **12** | **1.38** | **366.5** | **11** | **1.35** | **855.4** | **11** | **1.32** | **1096.4** |
| 3 | 15 | 3 | 3 | 9.06 | 17.4 | 3 | 10.80 | 11.7 | 3 | 3.48 | 29.0 |
| | [36,37] | 3 | 3 | 4.84 | 46.3 | 3 | 6.29 | 56.6 | 3 | 3.97 | 97.4 |
| | [70,75] | 3 | 3 | 6.12 | 251.2 | 3 | 5.84 | 331.0 | 3 | 6.23 | 1057.6 |
| | [138,140] | 3 | 3 | 4.52 | 1469.6 | 2 | 3.34 | 3927.21 | 2 | 4.21 | 4156.8 |
| | | **12** | **12** | **6.14** | **446.1** | **11** | **6.57** | **1081.6** | **11** | **4.47** | **1335.2** |
| 4 | 15 | 3 | 3 | 11.78 | 31.8 | 3 | 16.83 | 24.4 | 3 | 4.45 | 68.4 |
| | [36,37] | 3 | 3 | 15.31 | 158.0 | 3 | 16.32 | 205.9 | 3 | 12.86 | 371.2 |
| | [70,75] | 3 | 3 | 10.05 | 1392.7 | 3 | 9.67 | 2157.8 | 3 | 9.15 | 3341.0 |
| | [138,140] | 3 | 1 | 6.74 | 5481.5 | 1 | 6.27 | 6669.7 | 2 | 6.64 | 5945.3 |
| | | **12** | **10** | **10.97** | **1766.0** | **10** | **12.27** | **2264.5** | **11** | **8.27** | **2431.5** |
| 5 | 15 | 1 | 1 | 13.34 | 66.7 | 1 | 26.37 | 56.0 | 1 | 7.50 | 102.2 |
| | [36,37] | 1 | 1 | 32.74 | 542.6 | 1 | 30.94 | 427.7 | 1 | 23.32 | 855.1 |
| | [70,75] | 3 | 3 | 14.65 | 2265.3 | 2 | 13.82 | 4881.7 | 1 | 13.08 | 5785.8 |
| | [138,140] | 3 | 0 | 16.21 | 7200.0 | 0 | 15.36 | 7200.0 | 0 | 15.96 | 7200.0 |
| | | **8** | **5** | **17.33** | **3625.7** | **4** | **18.11** | **4591.1** | **3** | **14.74** | **4989.3** |
| TOTAL | | **44** | **39** | **8.19** | **1362.5** | **36** | **8.80** | **1980.6** | **36** | **6.52** | **2233.5** |

Table 7.11: Results for the Random75 instances

| | | | *Algorithm 2* (Ávila et al. (2017)) | | | *Algorithm 1* | | | *Algorithm 0* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Veh | $\|\mathbb{H}\|$ | inst | # opt | Gap0(%) | Time | # opt | Gap0(%) | Time | # opt | Gap0(%) | Time |
| | 18 | 6 | 6 | 3.54 | 34.2 | 6 | 4.03 | 15.2 | 6 | 1.76 | 58.0 |
| | 21 | 6 | 6 | 1.22 | 16.8 | 6 | 2.01 | 7.1 | 6 | 0.46 | 41.9 |
| 2 | 28 | 6 | 6 | 1.97 | 32.2 | 6 | 2.64 | 18.8 | 6 | 1.38 | 152.7 |
| | 33 | 6 | 6 | 1.15 | 36.1 | 6 | 2.57 | 47.9 | 6 | 1.24 | 114.1 |
| | | **24** | **24** | **1.97** | **29.8** | **24** | **2.82** | **22.2** | **24** | **1.21** | **91.7** |
| | 18 | 6 | 6 | 8.45 | 61.7 | 6 | 8.94 | 25.1 | 6 | 5.49 | 73.8 |
| | 21 | 6 | 6 | 7.35 | 42.6 | 6 | 8.02 | 27.4 | 6 | 5.73 | 67.1 |
| 3 | 28 | 6 | 6 | 2.99 | 90.5 | 6 | 2.51 | 44.0 | 6 | 2.46 | 145.3 |
| | 33 | 6 | 6 | 4.75 | 170.1 | 6 | 4.61 | 78.6 | 6 | 4.13 | 270.4 |
| | | **24** | **24** | **5.89** | **91.2** | **24** | **6.02** | **43.8** | **24** | **4.45** | **139.2** |
| | 18 | 4 | 4 | 12.81 | 73.0 | 4 | 13.20 | 30.9 | 4 | 9.79 | 78.1 |
| | 21 | 5 | 5 | 17.54 | 41.5 | 5 | 17.17 | 26.1 | 5 | 5.99 | 68.9 |
| 4 | 28 | 6 | 6 | 12.62 | 336.7 | 6 | 13.28 | 177.5 | 6 | 12.90 | 1790.2 |
| | 33 | 6 | 6 | 11.33 | 307.2 | 6 | 10.68 | 135.0 | 6 | 10.54 | 861.1 |
| | | **21** | **21** | **13.46** | **207.8** | **21** | **13.45** | **101.4** | **21** | **9.99** | **788.8** |
| | 18 | 3 | 3 | 12.67 | 85.2 | 3 | 18.84 | 47.5 | 3 | 4.14 | 123.2 |
| | 21 | 5 | 5 | 24.55 | 68.6 | 5 | 26.65 | 69.4 | 5 | 7.52 | 86.8 |
| 5 | 28 | 6 | 6 | 18.30 | 557.8 | 6 | 23.03 | 357.1 | 6 | 16.52 | 647.2 |
| | 33 | 3 | 3 | 15.24 | 411.9 | 3 | 14.83 | 404.2 | 3 | 13.23 | 398.9 |
| | | **17** | **17** | **18.61** | **304.8** | **17** | **21.91** | **226.2** | **17** | **11.11** | **346.1** |
| TOTAL | | **86** | **86** | **9.16** | **144.8** | **86** | **10.08** | **87.9** | **86** | **6.22** | **325.4** |

Table 7.12: Results for the Albaida instances

| | | | *Algorithm 2* (Ávila et al. (2017)) | | | *Algorithm 1* | | | *Algorithm 0* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Veh | $\|\mathbb{H}\|$ | inst | # opt | Gap0(%) | Time | # opt | Gap0(%) | Time | # opt | Gap0(%) | Time |
| | 22 | 6 | 6 | 1.55 | 130.3 | 6 | 2.93 | 93.8 | 6 | 0.68 | 253.7 |
| | 28 | 6 | 6 | 1.90 | 221.5 | 6 | 3.51 | 146.3 | 6 | 2.01 | 423.5 |
| 2 | 42 | 6 | 6 | 1.23 | 229.4 | 6 | 1.29 | 166.1 | 6 | 1.10 | 519.5 |
| | 47 | 6 | 6 | 1.20 | 248.5 | 6 | 1.16 | 170.6 | 6 | 1.28 | 573.6 |
| | | **24** | **24** | **1.47** | **207.4** | **24** | **2.22** | **144.2** | **24** | **1.27** | **442.6** |
| | 22 | 6 | 6 | 10.61 | 337.9 | 6 | 11.96 | 182.7 | 6 | 8.10 | 421.2 |
| | 28 | 6 | 6 | 9.67 | 398.3 | 6 | 8.92 | 257.5 | 6 | 8.10 | 558.1 |
| 3 | 42 | 6 | 6 | 4.61 | 1659.2 | 6 | 5.42 | 1429.0 | 6 | 4.55 | 1736.8 |
| | 47 | 6 | 5 | 4.60 | 2476.0 | 6 | 4.04 | 1766.5 | 6 | 4.59 | 1887.9 |
| | | **24** | **23** | **7.37** | **1217.9** | **24** | **7.59** | **908.9** | **24** | **6.34** | **1151.0** |
| | 22 | 5 | 5 | 22.93 | 1389.9 | 5 | 26.80 | 492.6 | 5 | 13.24 | 1467.6 |
| | 28 | 5 | 5 | 10.31 | 1107.5 | 5 | 9.98 | 590.9 | 5 | 8.52 | 1057.1 |
| 4 | 42 | 6 | 3 | 9.99 | 5616.0 | 6 | 8.65 | 2167.2 | 4 | 8.50 | 3709.0 |
| | 47 | 6 | 2 | 9.89 | 5397.7 | 4 | 8.65 | 4896.8 | 3 | 9.49 | 5563.5 |
| | | **22** | **15** | **12.98** | **3571.3** | **20** | **13.08** | **2172.6** | **17** | **9.85** | **3102.7** |
| | 22 | 2 | 2 | 21.67 | 1216.5 | 2 | 22.99 | 1139.8 | 2 | 20.62 | 1974.8 |
| | 28 | 2 | 2 | 21.56 | 1894.2 | 2 | 20.55 | 733.9 | 2 | 15.02 | 998.9 |
| 5 | 42 | 6 | 1 | 12.98 | 6309.6 | 4 | 12.80 | 4491.2 | 3 | 12.84 | 5798.9 |
| | 47 | 6 | 2 | 14.65 | 6110.9 | 2 | 12.98 | 5449.3 | 2 | 13.93 | 6177.7 |
| | | **16** | **7** | **15.77** | **5046.5** | **10** | **15.11** | **3961.9** | **9** | **14.49** | **4862.9** |
| TOTAL | | **86** | **69** | **8.72** | **2250.3** | **78** | **8.89** | **1586.8** | **74** | **7.34** | **2143.2** |

Table 7.13: Results for the Madrigueras instances

## 7.6 Conclusions

In this chapter we study the Distance-Constrained Close Enough Arc Routing Problem, which generalizes the Close Enough Arc Routing Problem to the case in which there is a fleet of vehicles based on a depot that jointly service a set of customers. Each customer is associated with a set of arcs which are close enough to it, such that the customer can be serviced by traversing any of these arcs. The length of the routes is limited by a given value and the objective is to minimize the sum of the route distances. The DC-CEARP is inspired by and has application to meter reading problems.

The contribution of this contribution is threefold. First, we propose a new formulation for the DC-CEARP that combines the best features of the previously existing ones. For this formulation, an exhaustive study of its associated polyhedron is performed, and several different families of valid inequalities are proposed. Secondly, many of the new inequalities presented here can be used, directly or easily adapted, in other arc routing problems, and the ideas in which some of the algorithms designed for the separation of these inequalities are based (or the algorithms themselves), can be used for similar inequalities in other problems. Finally, the designed branch-and-cut algorithm is an efficient exact method that is able to solve instances with up to 140 customers, 196 vertices, 544 arcs, and 5 vehicles to optimality within two hours computing time.

# Chapter 8

# On the Min-Max Close-Enough Arc Routing Problem

## 8.1 Introduction

We introduce a Close-Enough Arc Routing Problem where a fleet of homogeneous vehicles has to service a set of customers in such a way that the lengths of the routes are balanced. Each customer is associated with a subset of "close-enough" arcs and by traversing any of these arcs the vehicle services the customer. The problem, called the Min-Max Close-Enough Arc Routing Problem (MM-CEARP), consists of finding a set of routes for the vehicles, all of them starting and ending at the depot, jointly servicing all the customers, and such that the length of the longest route is minimized. The MM-CEARP is NP-*hard* as it generalizes the CEARP.

Min-max objectives are quite common in routing problems because minimizing the length of the longest route tends to balance the length or cost of the planned routes. Moreover, if the travel times are proportional to the travel distance, the last customer serviced is serviced as early as possible. The min-max objective for several arc routing problems was first proposed in Frederickson et al. (1978). About 20 years later, Applegate et al. (2002) considered a min-max problem in a newspaper delivery context. Since then, some other articles on arc routing problems have considered such an objective. The chapter by Benavent et al. (2014) summarizes the results obtained for some important min-max arc routing problems.

The main contribution is to introduce in the literature the MM-CEARP, focusing on its modeling and its exact solution. More precisely, we propose two different models for the problem: an arc-based formulation making use of arc and servicing variables, and

a route-based set covering formulation. Then, on the basis of the proposed models, we present a branch-and-cut algorithm as well as a branch-and-price algorithm. As for the B&P algorithm, an additional contribution comes from the definition of the first-level rule used in the branching scheme. In the route-based formulation, the min-max objective function characterizing the MM-CEARP forces to distinguish among the sets of feasible routes associated with the vehicles. This happens even if the sets are identical. The proposed branching scheme allows to recover integer solutions at the expenses of a diversification of the sets of feasible routes. Nevertheless, the first-level rule does not introduce symmetries in the solution space, does not alter the structure of the pricing problems, and, finally, allows the pricing problems to continue sharing the same feasible region. In turn, as long as only this rule is applied, this allows to design the sequential solution of the pricing problems (at each the column generation iteration) to potentially avoid solving some of them. Furthermore, the first-level rule consists of an application of the Ryan and Foster's branching rule (see Ryan and Foster (1981)), which is itself something not typical when (i) columns of the master program refer to elements of distinct sets and/or (ii) the B&P algorithm is addressing a routing problem. In particular, as for (ii), we have been able to efficiently handle the implications arising from the application of such a kind of rule thanks to the B&C algorithm used to solve the pricing problems to optimality. Again something not typical for B&P algorithms addressing routing problems, where the leading technique used to solve the pricing problems consists of dynamic programming algorithms.

The rest of the chapter is organized as follows. In Section 8.2, we formally define the MM-CEARP and present for the problem an arc-based and a route-based formulation. Solution algorithms to address the problem are then presented. In Section 8.3 we present a B&C algorithm addressing the arc-based formulation, whereas in Section 8.4 we describe a B&P algorithm based on the set covering formulation. A heuristic used to compute solutions with which initializing the exact algorithms is described in Section 8.5. The exact algorithms are compared in benchmark instances through extensive computational experiments in Section 8.6. Finally, conclusions are drawn in Section 8.7.

## 8.2 Problem definition and formulation

Let $G = (V, A)$ be a strongly connected directed graph with set of vertices $V$, where vertex 1 denotes the depot, and set of arcs $A$, and let $d_{ij} \geq 0$ be the an integer value representing the length/distance associated with the traversal of arc $(i, j) \in A$. There is a fleet of $K$ identical vehicles based at the depot and a set of $L$ customers. Each customer $c \in \{1, \ldots, L\}$ has associated a set of arcs $H_c \subseteq A$ from which it can be

served. We consider that a customer $c$ is served if there is a vehicle $k$ that traverses at least one arc in $H_c$. Note that the subsets $H_c$ do not need to be disjoint nor induce connected subgraphs. The Min-Max Close-Enough Arc Routing Problem consists of finding a set of $K$ routes, starting and ending at the depot, servicing all the customers and minimizing the length of the largest route.

In what follows, $\mathbb{K} = \{1, \ldots, K\}$ will represent the set of vehicles and $\mathbb{H} = \{1, \ldots, L\}$ the set of customers. Given sets $S, S_1, S_2 \subset V$, we define $(S_1, S_2) = \{(i,j) \in A : i \in S_1\ j \in S_2\}$, $\delta^+(S) = (S, V \setminus S)$, $\delta^-(S) = (V \setminus S, S)$, $\delta(S) = \delta^+(S) \cup \delta^-(S)$, and $A(S) = \{(i,j) \in A : i, j \in S\}$. Finally, given a vector $x$ indexed on the arcs, and given a set $F$ of arcs, $x(F) = \sum_{(i,j)\in F} x_{ij}$.

### 8.2.1 Arc-based formulation

In this section we present an ILP formulation for the MM-CEARP, very similar to one of the four proposed by Ávila et al. (2017) for the DC-CEARP, which uses an artificial variable $w$ to model the minimization of the maximum length route and the following two sets of variables:

$x_{ij}^k$ = number of times that the vehicle $k$ traverses arc $(i,j) \in A$,

$$z_c^k = \begin{cases} 1, & \text{if customer } c \text{ is served by vehicle } k \\ 0, & \text{otherwise.} \end{cases}$$

The first MM-CEARP formulation is:

$$\text{Minimize} \qquad w$$

s.t.:

$$\sum_{k\in\mathbb{K}} z_c^k = 1 \qquad \forall c \in \mathbb{H} \tag{8.1a}$$

$$\sum_{(i,j)\in A} d_{ij} x_{ij}^k \leq w \qquad \forall k \in \mathbb{K} \tag{8.1b}$$

$$x^k(\delta^+(i)) = x^k(\delta^-(i)) \qquad \forall\, i \in V,\ \forall k \in \mathbb{K} \tag{8.1c}$$

$$\sum_{(i,j)\in H_c} x_{ij}^k \geq z_c^k \qquad \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{8.1d}$$

$$x^k(\delta^+(S)) \geq z_c^k - x^k(H_c \cap A(V \setminus S)) \qquad \forall S \subset V \setminus \{1\},\ \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{8.1e}$$

$$x_{ij}^k \geq 0 \text{ and integer} \qquad \forall (i,j) \in A,\ \forall k \in \mathbb{K} \tag{8.1f}$$

$$z_c^k \in \{0,1\} \qquad \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{8.1g}$$

Equations (8.1a) force the service of all customers while inequalities (8.1b) imply that the length of any route is less than or equal to $w$, and, together to the objective function, that the length of the longest route is minimized. Constraints (8.1c) are the well known symmetry equations for each vertex in $V$. Inequalities (8.1d) ensure that if a vehicle serves a customer $c$, at least one arc in $H_c$ must be traversed. The connectivity of each route is guaranteed by inequalities (8.1e). If vehicle $k$ does not serve customer $c$, $z_c^k = 0$ and the inequality is trivially satisfied. Otherwise, if vehicle $k$ serves customer $c$ by traversing an arc in $H_c \cap A(V \setminus S)$, then it does not need to traverse the cut-set $\delta(S)$ and the inequality is also trivially satisfied. Only when vehicle $k$ serves customer $c$ by traversing an arc not in $H_c \cap A(V \setminus S)$ (hence, traversing an arc in $\delta(S)$ or in $A(S)$), the vehicle has to traverse $\delta(S)$ and, therefore, the inequality is satisfied. Note that there is an exponential number of such inequalities. Finally, (8.1f)–(8.1g) define the domain of the variables.

In what follows we present some inequalities proposed in Ávila et al. (2017) and Corberán et al. (2021) for the DC-CEARP. They are also valid for our problem and will be used to strengthen the linear relaxation of the above formulation.

**Parity inequalities**

Parity inequalities are implied by the fact that any cutset has to be traversed by each vehicle an even, or zero, number of times. Note that symmetry equations (8.1c) guarantee that every node has even degree in the graph induced by any integer solution $x \in \mathbb{Z}^{|A|}$. However, if $x$ is fractional, this is not necessarily true and, therefore, parity inequalities can help to cut this kind of "solutions".

Let $S \subseteq V \setminus \{1\}$ and $F^{\mathbb{H}} = \{c_1, c_2, ..., c_q\}$, where $q \geq 3$ and odd, satisfying

- $H_{c_i} \cap H_{c_j} \cap \delta(S) = \emptyset$ and
- $H_{c_i} \cap \delta(S) \neq \emptyset \quad \forall i \in \{1, \ldots, q\}$.

The following inequality is called *disaggregate z-parity inequality*, because is associated with a single vehicle $k$, and is valid for the MM-CEARP

$$x^k(\delta(S)) \geq \sum_{i=1}^{q} \Big( 2z_{c_i}^k - 1 - 2x^k \left( H_{c_i} \setminus \delta(S) \right) \Big) + 1. \tag{8.2a}$$

Basically, the inequality establishes that if vehicle $k$ serves customer $c_i$ and does not traverse any edge in $H_{c_i} \setminus \delta(S)$, then $k$ serves $c_i$ by traversing at least an arc in $H_{c_i} \cap \delta(S)$. If this is true for the $q$ customers in $F^{\mathbb{H}}$, vehicle $k$ has to traverse $\delta(S)$ at least $q$ times, and, since $q$ is odd, $k$ has to traverse the cutset at least one more time.

Parity inequalities can be generalized to any subset of vehicles as follows. Given a subset of vehicles $\Omega = \{k_1, .., k_p\}$, the associated $\Omega$-*aggregate z-parity inequality* is

$$\sum_{k\in\Omega} x^k(\delta(S)) \geq \sum_{i=1}^{q}\Big(\sum_{k\in\Omega} 2z^k_{c_i} - 1 - 2\sum_{k\in\Omega} x^k\left(H_{c_i} \setminus \delta(S)\right)\Big) + 1. \tag{8.2b}$$

If $\Omega = \mathbb{K}$, we have the *aggregate parity inequality*

$$\sum_{k\in\mathbb{K}} x^k(\delta(S)) \geq \sum_{i=1}^{q}\Big(1 - 2\sum_{k\in\mathbb{K}} x^k\left(H_{c_i} \setminus \delta(S)\right)\Big) + 1. \tag{8.2c}$$

**Max-distance constraints**

Let $w^*$ be an upper bound to the length of the longest route. Let $F^{\mathbb{H}} \subseteq \mathbb{H}$ be a subset of customers such that the value of the tour of minimal length servicing $F^{\mathbb{H}}$ (or a lower bound of it) is greater than $w^*$. In this case, that tour cannot be optimal for the MM-CEARP and therefore all the customers in $F^{\mathbb{H}}$ cannot be served by a single vehicle in an optimal solution. Let $S$ be the set of vertices incident with the arcs in $\cup_{c\in F^{\mathbb{H}}} H_c$ and suppose $1 \notin S$. By the previous reasoning we have that at least two different vehicles must enter $S$ and the following inequality must be satisfied by any optimal MM-CEARP solution.

$$\sum_{k\in\mathbb{K}} x^k(\delta^-(S)) \geq 2. \tag{8.3a}$$

Note, however, that inequality (8.3a) allows the same vehicle to enter $S$ twice, while there must be two *different* vehicles entering $S$. Hence, we can add the inequalities:

$$\sum_{k\neq k'} x^k(\delta^-(S)) \geq 1, \quad \forall k' \in \mathbb{K}. \tag{8.3b}$$

Inequalities (8.3a) and (8.3b) can be extended in the following way. Let $v(F^{\mathbb{H}}) \geq 2$ be the minimum number of vehicles needed to serve the customers in $F^{\mathbb{H}}$ (or a lower bound). For any subset of vehicles $\Omega \subseteq \mathbb{K}$, with $0 \leq |\Omega| \leq v(F^{\mathbb{H}}) - 1$ we have the following inequality:

$$\sum_{k\in\mathbb{K}\setminus\Omega} x^k(\delta^-(S)) \geq v(F^{\mathbb{H}}) - |\Omega|. \tag{8.3c}$$

Furthermore, if a single vehicle cannot serve all the customers in $F^{\mathbb{H}}$ in an optimal solution, we can add the following inequality for any vehicle $k \in \mathbb{K}$:

$$z_c^k(F^{\mathbb{H}}) \leq |F^{\mathbb{H}}| - 1, \tag{8.3d}$$

which, as above, can be generalized if $v(F^{\mathbb{H}}) \geq 2$. For any subset of vehicles $\Omega \subseteq \mathbb{K}$, with $1 \leq |\Omega| \leq v(F^{\mathbb{H}}) - 1$ we have the following inequality:

$$\sum_{k \in \Omega} z_c^k(F^{\mathbb{H}}) \leq |F^{\mathbb{H}}| - (v(F^{\mathbb{H}}) - |\Omega|). \tag{8.3e}$$

The value for $w^*$ can be obtained by using the heuristic described in Section 8.5, while $v(F^{\mathbb{H}})$ can be computed with the exact algorithm proposed in Corberán et al. (2021).

### 8.2.2 Route-based formulation

As illustrated in Barnhart et al. (1998), most routing problems can be formulated in a natural way as set partitioning problems where the columns (of the coefficient matrix) correspond to feasible routes for the vehicles and each row (of the coefficient matrix) corresponds to the requirement that a customer must be served *exactly* once. Alternatively, the problem can be formulated as a set covering problem in which it is required that each customer is served *at least* once. Note that, if a subcolumn of a feasible column defines another feasible column with lower cost, an optimal solution to the set covering problem will define an optimal set partitioning solution and, hence, it is possible to work with any of the two formulations. However the set covering formulation has the following advantages:

- its linear programming relaxation is numerically more stable and thus easier to solve, and
- it is trivial to construct a feasible integer solution from a solution to the linear programming relaxation.

According to these insights, we modeled the MM-CEARP by means of a route-based set covering formulation that leads the bases for the B&P algorithm discussed in Section 8.4.

Let $R^k$ be the set of feasible routes for vehicle $k \in \mathbb{K}$. Feasibility takes into account constraints (8.1b)-(8.1g) for each vehicle $k \in \mathbb{K}$. For each $r \in R^k$, let $d^{kr}$ be the length of the route. Moreover, for each customer $c \in \mathbb{H}$ and each $r \in R^k$, let $s_c^{kr}$ be a binary parameter equal to 1 if the route $r$ serves customer $c$ and 0 otherwise.

Then, let's consider a set of variables associated with the use of the routes:

$$\lambda^{kr} = \begin{cases} 1, & \text{if the route } r \in R^k \text{ is assigned to the vehicle } k \in \mathbb{K}, \\ 0, & \text{otherwise,} \end{cases}$$

and another set of variables modeling the length of the route assigned to each vehicle:

$w^k$ = length of route assigned to vehicle $k \in \mathbb{K}$.

Using this notation, the MM-CEARP can be formulated as follows:

$$\text{Minimize} \qquad w^1$$

s.t.:

$$\sum_{k \in \mathbb{K}} \sum_{r \in R^k} s_c^{kr} \lambda^{kr} \geq 1 \qquad \forall c \in \mathbb{H} \tag{8.4a}$$

$$\sum_{r \in R^k} \lambda^{kr} = 1 \qquad \forall k \in \mathbb{K} \tag{8.4b}$$

$$\sum_{r \in R^k} d^{kr} \lambda^{kr} - w^k \leq 0 \qquad \forall k \in \mathbb{K} \tag{8.4c}$$

$$w^k - w^{k+1} \geq 0 \qquad \forall k = 1, \ldots, K-1 \tag{8.4d}$$

$$\lambda^{kr} \in \{0, 1\} \qquad \forall k \in \mathbb{K},\ \forall r \in R^k \tag{8.4e}$$

The objective function minimizes the length of the longest route. This is ensured by constraints (8.4c) together with (8.4d). Actually, constraints (8.4c) define the lengths of the routes assigned to the vehicles. Then, constraints (8.4d) impose the lengths of the routes associated with vehicles from 1 to $K$ to be sorted in non-increasing order. The mandatory service of the customers is established in inequalities (8.4a). The convexity constraints (8.4b) imply that a single route $r \in R^k$ is assigned to each vehicle $k \in \mathbb{K}$. Finally, constraints (8.4e) define the domain for the $\lambda^{kr}$ variables. Constraints $w^k \geq 0$ are implied by inequalities (8.4c).

Note that sets $R^k$, $k \in \mathbb{K}$, are all identical. Nevertheless, we decided to index them (by vehicle index) to have a notation allowing us to better explain the B&P algorithm (see Section 8.4). In particular, the reason for using such a notation will be clarified in Section 8.4.2.

## 8.3 Branch-and-cut algorithm

In this section, we describe the branch-and-cut algorithm for solving the MM-CEARP, which relies on the arc-based formulation presented in Section 8.2.1 and the use of mixed-integer programming (MIP) solver.

### 8.3.1 Separation algorithms

Here we describe the separation algorithms that have been used to identify inequalities that are violated by the current LP solution at any iteration of the cutting-plane phase of the branch-and-cut algorithm, which includes separation methods for identifying connectivity (8.1e), aggregated parity (8.2c), and max-distance((8.3a), (8.3b), and (8.3d)) violated inequalities.

#### Connectivity inequalities

To identify violated connectivity inequalities (8.1e) we have used a heuristic procedure proposed in Ávila et al. (2017) for the DC-CEARP. Given a solution $(x^{k*}, z^{k*})$ of the linear relaxation corresponding to a vehicle $k$, we first build the graph induced by the arcs $a \in A$ such that $x_a^{k*} \geq \varepsilon$, where $\varepsilon$ is a given parameter. If the support graph is not weakly connected, let $C_1, \ldots, C_q$ be its weakly connected components. For each $C_i$, let $S$ be its associated set of vertices. We look for the customer $c \in \mathbb{H}$ such that $z_c^{k*} - x^{k*}(H_c \cap A(V \setminus S)$ is maximized. If $x^{k*}(\delta^+(S)) < z_c^{k*} - x^{k*}(H_c \cap A(V \setminus S)$ the corresponding connectivity constraint (8.1e) is violated.

A second heuristic (described in Corberán et al. (2021)) based on the Gomory-Hu algorithm is also applied.

#### Parity inequalities

To separate parity inequalities (8.2c) we have implemented the following heuristic algorithm. Note that these inequalities can be written as

$$\sum_{k \in \mathbb{K}} x^k(\delta(S)) \geq \sum_{k \in \mathbb{K}} \sum_{i=1}^{q} \left( z_{c_i}^k - 2x^k\big(H_{c_i} \setminus \delta(S)\big)\right) + 1.$$

If $(x^{k*}, z^{k*})$ are the values of the variables associated with vehicle $k$ in the solution of the linear relaxation, let $(\bar{x}^*, \bar{z}^*)$ be the aggregated solution, that is, $\bar{x}_a^* = \sum_{k \in \mathbb{K}} x_a^{k*}$ and $\bar{z}_c^* = \sum_{k \in \mathbb{K}} z_c^{k*} = 1$. First, we create the graph induced by the arcs $a \in A_R = H_1 \cup \cdots \cup H_L$ with $\bar{x}_a^* \geq 1 + \varepsilon$ and by the arcs $a \notin A_R$ with $\bar{x}_a^* > \varepsilon$, where $\varepsilon$

is a given parameter. Let $C_1, \ldots, C_k$ be the weakly connected components of this graph. Then, given a connected component $C_i$ and its associated set of vertices $S$, we compute $\bar{x}^*(\delta(S) \cap A_R)$ and check if this value is close to an odd number, that is, $2n + 0.75 \leq \bar{x}^*(\delta(S) \cap A_R) \leq 2n + 1.25$. If so, the heuristic tries to select $q = 2n + 1$ customers among those having arcs in the cutset in order to form set $F^{\mathbb{H}}$ as described in Section 8.2.1. To do so, we iteratively add customers to $F^{\mathbb{H}}$ in decreasing order of the $\bar{z}_c^* - 2\bar{x}^*(H_c \setminus \delta(S))$ values, such that the sets $H_c \cap \delta(S)$ are disjoint with those associated with the previously selected customers, until we reach the desired number $q$ of customers. If there are not enough customers that can be selected, we choose another component. Otherwise, we check if the inequality (8.2c) is violated.

#### Max-distance inequalities

In order to find violated max-distance inequalities (8.3a) and (8.3b), we use the heuristic separation algorithm described in Corberán et al. (2021) and denoted as A11. Inequalities (8.3d) are also separated heuristically using the algorithm proposed in Corberán et al. (2021) and denoted as A12.

### 8.3.2 Initial relaxation and cutting-plane algorithm

The initial LP relaxation contains all the inequalities in the formulation except for the connectivity inequalities, which are exponential in number. At each cutting plane iteration, the separation algorithms are applied in the following order:

1. Connectivity inequalities separation algorithm based on connected components with $\epsilon = 0, 0.25, 0.5, 0.75$.
2. Connectivity inequalities separation algorithm based on Gomory-Hu.
3. Only at the root node, parity inequalities separation algorithm with $\epsilon = 0, 0.25, 0.5, 0.75$.
4. Only at the root node, max-distance inequalities separation algorithms.

This cutting-plane algorithm is applied at each node of the tree until no new violated inequalities are found. When this happens, we branch using the strong branching strategy provided by the MIP solver. This strategy branches on variables and allows to assign different priorities to them. Variables with higher priority are the first ones used for branching. We have assigned a higher priority to the $z_c^k$ variables.

## 8.4 Branch-and-price algorithm

When a set covering problem is addressed by means of a B&P algorithm, its formulation, in our case formulation (8.4), is usually referred as master program (MP). In the B&P algorithm, at each node of the branch-and-bound tree, the linear relaxation of the MP (LMP), eventually augmented by branching constraints, is solved iteratively by means of column generation. The starting point is to define the LMP over a subset $\tilde{R} \subseteq \bigcup_{k \in \mathbb{K}} R^k$ of the feasible routes for the vehicles. This restricted version of LMP is usually called reduced linear master program (RLMP). At each iteration, column generation alternates between the optimization of the RLMP and the solution of pricing problems (PPs). The former allows to retrieve optimal dual variable values with respect to set $\tilde{R}$. The latter, on the bases of the dual variable values, generates negative reduced cost route variables $\lambda^{kr}$ to be included in the RLMP, if any. When no negative reduced cost variable is found, the optimal solution of the RLMP is also the optimal solution of the LMP (Desaulniers et al. (2005)). Branching is finally required to ensure the integrality of the solution.

### 8.4.1 Column generation

Let us consider the linear relaxation of (8.4) at the root node of the branch-and-bound tree. The dual variables associated with the constraints (8.4a), (8.4b), (8.4c), and (8.4d) are respectively:

- $\mu_c \in \mathbb{R}^+$, for each customer $c \in \mathbb{H}$,
- $\theta_k \in \mathbb{R}$, for each vehicle $k \in \mathbb{K}$,
- $\rho_k \in \mathbb{R}^-$, for each vehicle $k \in \mathbb{K}$,
- $\sigma_k \in \mathbb{R}^+$, for each $k = 1, \ldots, K-1$.

Using these dual variables in their respective domain, we are able to express the formulation of the dual of LMP as follows:

$$\max \sum_{c \in \mathbb{H}} 1 \cdot \mu_c + \sum_{k \in \mathbb{K}} 1 \cdot \theta_k + \sum_{k \in \mathbb{K}} 0 \cdot \rho_k + \sum_{k=1}^{K-1} 0 \cdot \sigma_k \tag{8.5a}$$

$$s.t. \sum_{c \in \mathbb{H}} s_c^{kr} \mu_c + \theta_k + d^{kr} \rho_k \leq 0 \qquad \forall k \in \mathbb{K},\ \forall r \in R^k \tag{8.5b}$$

$$-\rho_1 + \sigma_1 \leq 1 \tag{8.5c}$$

$$-\rho_k + \sigma_k - \sigma_{k-1} \leq 0 \qquad \forall k = 2, \ldots K-2 \tag{8.5d}$$

$$-\rho_K - \sigma_{K-1} \leq 0 \tag{8.5e}$$

where there is a constraint (8.5b) for each variable $\lambda^{kr}$ of the primal formulation, and constraints (8.5c)– (8.5e) are related with each $w^k$ variable, $k \in \mathbb{K}$.

Thus, based on the dual formulation (8.5), we can see that there is one distinct PP for each vehicle $k \in \mathbb{K}$. In particular, given the duals $(\boldsymbol{\mu}, \boldsymbol{\theta}, \boldsymbol{\rho}, \boldsymbol{\sigma})$, the PP for vehicle $k \in \mathbb{K}$ consists of finding a minimum reduced cost route to be assigned to the vehicle, where the reduced cost $\bar{c}^{kr}(\boldsymbol{\mu}, \boldsymbol{\theta}, \boldsymbol{\rho})$ of route $r \in R^k$ to be assigned to the vehicle is defined as:

$$\bar{c}^{kr}(\boldsymbol{\mu}, \boldsymbol{\theta}, \boldsymbol{\rho}) = -\sum_{c \in \mathbb{H}} s_c^{kr} \mu_c - \theta_k - d^{kr} \rho_k \tag{8.6}$$

A solution (a route) corresponds to a negative reduced cost $\lambda^{kr}$ variable if its value (reduced cost) is less than 0.

### Pricing problem modeling

In order to define the pricing problem, we consider the same variables (with the same meaning) as those used in formulation (8.1). The PP associated with vehicle $k \in \mathbb{K}$ can then be formulated as follows:

$$\text{Minimize } -\sum_{c \in \mathbb{H}} \mu_c z_c^k - \sum_{(i,j) \in A} d_{ij} x_{ij}^k \rho_k.$$

s.t.:

$$x^k(\delta^+(i)) = x^k(\delta^-(i)) \qquad \forall\, i \in V \tag{8.7a}$$

$$x^k(\delta^+(S)) \geq z_c^k - x^k(H_c \cap A(V \setminus S)) \qquad \forall S \subset V \setminus \{1\},\ \forall c \in \mathbb{H} \tag{8.7b}$$

$$\sum_{(i,j) \in H_c} x_{ij}^k \geq z_c^k \qquad \forall c \in \mathbb{H} \tag{8.7c}$$

$$x_{ij}^k \geq 0 \text{ and integer} \qquad \forall (i,j) \in A \tag{8.7d}$$

$$z_c^k \in \{0,1\} \qquad c \in \mathbb{H}, \tag{8.7e}$$

where $\mu_c \geq 0$, $\theta_k \in \mathbb{R}$, $\rho_k \leq 0$ and, hence, $-d_{ij}\rho_k \geq 0$ for each $(i,j) \in A$. The objective function aims at minimizing the reduced cost of the route. Constraints (8.7a) are the symmetry equations for each vertex, while constraints (8.7b) are used to ensure the connectivity of the optimal solution. Consistency between the $x_{ij}^k$ and $z_c^k$ variables is imposed through constraints (8.7c).

An optimal solution to (8.7) corresponds to a negative reduced cost variable if its value is less than $\theta_k$.

Moreover, when an upper bound $W$ is available for $w^1$, $R^k$ can be restricted to include feasible routes such that $d^{kr} \leq W - 1$ ($d_{ij}$ is an integer value for each $(i, j) \in A$), and we can include in formulation (8.7) the following constraint:

$$\sum_{(i,j)\in A} d_{ij} x_{ij}^k \leq W - 1. \tag{8.7f}$$

Note that the valid inequalities (8.7a) - (8.7e), which defines the feasible region of the PP associated with vehicle $k \in \mathbb{K}$, are the same as the inequalities (8.1c) - (8.1g) appearing in formulation (8.1) for each vehicle $k \in \mathbb{K}$. Thus, the disaggregate $z$-parity inequalities (8.2a) are also valid for model (8.7).

**A branch-and-cut algorithm for the pricing problem**

In **?**, the authors introduce the *Profitable Close-Enough Arc Routing Problem* (PCEARP). Let $G = (V, A)$ be a directed and strongly connected graph with a cost $c_{ij} \geq 0$ associated with each arc $(i, j) \in A$ and a distinguished vertex 1 as the depot. Let $\mathbb{H}$ be the set of customers, each of them has an associated set of arcs $H_c \subseteq A$ in such a way a customer $c$ is served when at least one of the arcs in $H_c$ is traversed. Associated with each customer $c$ there is a profit $p_c \geq 0$ that is collected (only once) if the customer is served. The PCEARP consists of finding a tour starting and ending at the depot and maximizing the difference between the total profit collected and the cost of the route. Therefore, for each vehicle $k \in \mathbb{K}$, the pricing problem can be seen as a PCEARP with the additional constraint (8.7f). In fact, it is possible to rewrite the objective function as a maximization problem with $p_c = \mu_c \geq 0$ and $c_{ij} = -d_{ij}\rho_k \geq 0$.

Finally, it is worth observing that all the PPs share the same feasible region at the root node of the branch-and-bound tree. However, as will be explained in Section 8.4.2, branching rules may differentiate the pricing problem feasible regions in the subtree arising from their application.

We solve the pricing problem by using a branch-and-cut algorithm similar to the one described in Bianchessi et al. (2021) for solving the PCEARP.

When solving the pricing problem, it may be advantageous to save as many routes (columns) as we can find. Therefore, every time that the branch-and-cut algorithm finds an integer solution with negative reduced cost, we store it in order to add it to the restricted master problem. Furthermore, for each stored route, we study if it traverses any arc $a \in H_c$ associated with a customer $c$ having $\mu_c = 0$. If this happens, we mark this customer as served by the route.

### Initial relaxation

The initial relaxation includes all the sets of constraints (8.7a)-(8.7f). In particular, let $S_c$ be the set of vertices incident with the arcs in $H_c$, $c \in \mathbb{H}$. In the initial relaxation are included only connectivity constraints (8.7b) associated with sets $S_c$ such that $1 \notin S_c$.

Moreover, in order to obtain routes useful for the LMP, inequalities (8.7g) and (8.7g) are also included in the initial relaxation. Inequality (8.7g) forces the vehicle to leave the depot, while inequality (8.7h) ensures that at least one customer will be served by the vehicle.

$$x^k(\delta^+(1)) \geq 1 \tag{8.7g}$$

$$\sum_{c=1,\ldots,L} z_c^k \geq 1. \tag{8.7h}$$

In order to strengthen this linear relaxation, we also add some max-distance constraints (8.3d) associated with sets of two and three customers. To identify these sets for each pair of customers $c_i$ and $c_j$, we calculate a lower bound on the cost of servicing them with the same vehicle as the sum of the following costs: the cost of the shortest path from the depot to any arc in $H_{c_i}$, the cost of the shortest arc in $H_{c_i}$, the minimum cost to go from any arc in $H_{c_i}$ to any arc in $H_{c_j}$, the cost of the the shortest arc in $H_{c_j}$, and, finally, the cost of the shortest path between any arc in $H_{c_j}$ and the depot. In addition, as we are in a directed graph, we also have to consider the case in which $c_j$ is served before than $c_i$. If the cost of both routes exceeds $W$, the two customers cannot be served by the same route and we add the corresponding max-distance constraint (8.3d) to the initial relaxation. A similar bound can be computed for sets of three customers.

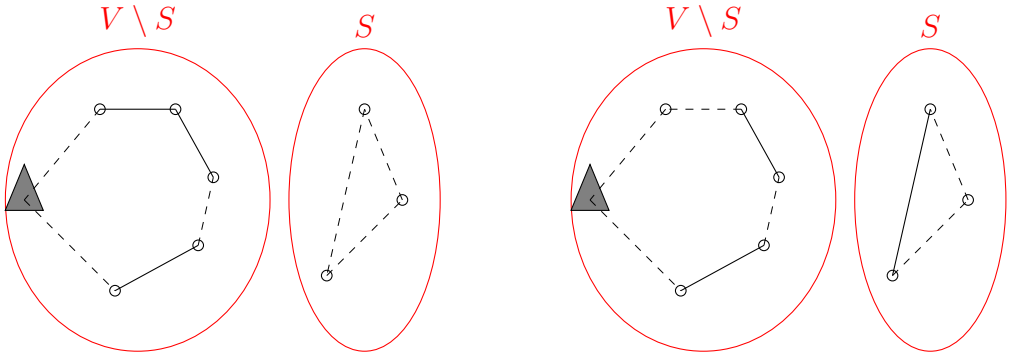
### Separation algorithms

In the branch-and-cut algorithm for the pricing problem we separate connectivity (8.7b) and parity (8.2a) inequalities. The separation algorithm used for parity inequalities is similar as the one described in Section 8.3.1 but without aggregating the solution.

For the connectivity inequalities we apply the first separation heuristic described in Section 8.3.1 with a modification that will be described in what follows.

Note that constraints (8.7b) do not guarantee that all solutions of the formulation will be connected, since there are still two situations in which disconnected subtours

may appear. The first one is if $\rho_k = 0$. In this case, a solution can contain cycles with arcs that do not belong to any served customer (see Figure 8.1a, where the triangle represents the depot and the solid lines represent arcs of a served customer), but these cycles can be removed without affecting the reduced cost of the solution. The other situation may occur when there is a cycle disconnected from the depot, but for any customer with $z_c^k = 1$ there is at least one arc in $H_c$ traversed and connected to the depot (see Figure 8.1b). But this solution will not be optimal, since this cycle can be removed from the solution while still servicing the same customers, thus decreasing the reduced cost of the route.




(a) subtour not traversing any served customer

(b) subtour traversing a served customer

Figure 8.1: Solutions of the PP with subtours that satisfy connectivity inequalities (8.7b)

However, as will be explained in Section 8.4.2, the branching rules of the branch-and-price procedure may introduce some lower bound on the use of some arcs, that is, on some $x_{ij}^k$ variables. If there is a disconnected subtour containing one of these arcs with a lower bound greater than 0, it is not possible to remove this cycle from the solution. For this reason, for each connected component $C_i$, if it does not contain the depot, we check if there is an arc incident with vertices in $C_i$ having a lower bound greater than 0. If such an arc is found, the inequality $x^k(\delta(C_i)) \geq 1$ is a valid inequality that is violated by this solution.

### Cutting-plane algorithm

The cutting-plane algorithm applies the following separation algorithms in the order in which are listed:

1. Heuristic separation algorithm for connectivity inequalities with $\varepsilon = 0,\ 0.25,\ 0.5,\ 0.75$.
2. Heuristic separation algorithm for parity inequalities with $\varepsilon = 0,\ 0.25,\ 0.5$ (only at the root node).

The cutting-plane algorithm is applied at each node of the tree until no new violated inequalities are found. Again, we branch using the MIP-solver implementation of the strong branching strategy by giving higher priority to the $z_c^k$ variables.

**Primal heuristic**

To obtain a higher number of columns and good lower bounds that can help reducing the size of the branch-and-cut search tree, we have implemented a heuristic algorithm using the fractional solutions of the LPs at the nodes of the tree.

Let $(x^*, z^*)$ be a fractional solution. The subset of arcs $A_R = H_1 \cup \cdots \cup H_L$ is split into four different subsets according to their $x^*$ value. The first subset includes arcs $a$ with $x_a^* \geq 0.9$. The second, those with $x_a^* \in [0.7, 0.9)$, the third the arcs with $x_a^* \in [0.5, 0.7)$, and the last subset includes the arcs $a \in A_R$ with $x_a^* < 0.5$, which will not be considered in the procedure.

We start by building a solution by iteratively selecting arcs from the first subset. For each arc $a$ of this subset, we calculate the profit obtained from traversing it, given by the sum of the profit $\mu_c$ of the customers $c$ not yet served such that $a \in H_c$. Then, the arc with the maximum profit is added to $\mathcal{A}$ and removed from its corresponding subset. Moreover, the customers served by traversing this arc are labeled as served. This procedure is repeated until the first subset is empty or there are no new customers that can be served traversing the remaining arcs. Then, we repeat the procedure with the second subset and, if necessary, with the third one.

Once the set $\mathcal{A}$ has been obtained, a route traversing this subset of arcs is built. The route is initialized by randomly selecting an arc in $\mathcal{A}$. Then, all the remaining arcs are allocated using a deterministic completion procedure. For each unassigned arc $a \in \mathcal{A}$, we compute the cost of inserting the arc in the route in the best possible position and add the one with the minimum insertion cost. We proceed until all the arcs in $\mathcal{A}$ are allocated. Once the route is complete, we check which customers are served. If the resulting solution improves the current LB, we stop. Otherwise, we solve a Directed General Routing Problem (DGRP) in which all the arcs in $\mathcal{A}$ are "required" and the depot is a "required vertex", using the exact procedure described in Ávila et al. (2015). The DGRP consists of finding a minimum cost route that traverses all the required arcs and visits all the required nodes at least once. As before, we study the customers served by the obtained route and check if it improves the current LB.

This algorithm is executed at every 100 iterations of the cutting-plane procedure at the root node. Once the root node has been studied, it is executed once every 20 nodes up to node number 200, once every 50 nodes between nodes 201 and 501, and once every 200 nodes beyond that number.

### Solution of the PPs

Let $\mathcal{K} = \{v_1, \ldots, v_K\}$ be the set of vehicles sorted in non-ascending order with respect to their corresponding $\rho_{v_k}$ values. At each column generation iteration, the PPs are considered sequentially starting from the problem associated with vehicle $v_1$. Let $\mathcal{R}^{v_k}$ be the set of routes found by solving the PP for vehicle $v_k \in \mathcal{K}$. For each route $r \in \mathcal{R}^{v_k}$, we check if it corresponds to a negative reduced cost $\lambda^{v_k r}$ variable (column), meaning that we check if its cost is less than $\theta_{v_k}$.

Additionally, we check if the route $r$ corresponds to a negative reduced cost column for any of the other vehicles. In this way, as long as all the PPs share the same feasible region, we avoid to solve subsequent PPs corresponding to vehicles $v_t$, $t > k$, such that $|\rho_{v_k} - \rho_{v_t}| < \epsilon$, with $\epsilon \to 0^+$. This does not hold anymore once a branching rule which diversifies the pricing problem feasible regions is applied (see Section 8.4.2). When this happens, in each node of the subtree arising from the application of such a branching rule, all the PPs have to be eventually solved at each column generation iteration. The sequential solution of the PPs terminates as soon as negative reduced cost columns are found after solving one of them, or when all the PPs have been solved and no negative reduced cost column has been generated.

### Heuristic column generation

In many iterations of the column generation algorithm (especially in the initial ones), there are many routes with negative reduced cost that can be efficiently found by means of an heuristic algorithm. Hence, at each column generation iteration, the solution mechanism outlined in Section 8.4.1 is first applied considering a Greedy Randomized Adaptive Search Procedure (GRASP) as solver for the PPs. In case no negative reduced cost column is generated by means of the heuristic, the mechanism is reapplied by solving the PPs to optimality through the B&C algorithm described in Section 8.7.

The heuristic is initialized by computing an initial profit for each arc in $A_R = H_1 \cup \cdots \cup H_L$ as the cost of going from the depot to the arc and back minus the profit of serving all possible customers. Out of all these arcs, we define a subset $\bar{A}_R$ of $min\{50, \frac{|A_R|}{2}\}$ arcs including those associated with the minimum profit. At each iteration of the GRASP, a route is initialized by randomly selecting an arc in $\bar{A}_R$ and then, completed and improved, respectively, by using the *Construction* and *IteratedLocalSearch* subroutines described in the following.

**Construction.** Initial solutions are built using an adaptation of the path-scanning procedure, which selects the subset of customers associated with the smallest profits.

For each customer not yet assigned, the corresponding profit $\psi_c$ is computed as the difference in the reduced cost of the route if $c$ is served through the route. A restricted candidate list (RCL) is built including the customers candidates associated with the smallest profits and taking into account a threshold parameter $\alpha$. Given $\psi_{min}$ and $\psi_{max}$, a customer is included in the RCL if

$$\psi_c \geq \alpha(\psi_{max} - \psi_{min}) + \psi_{min}.$$

As usual, parameter $\alpha$ controls the greediness of the selection ($\alpha = 0$ pure greedy; $\alpha = 1$ pure random). Looking for a tiny randomization component within a greedy procedure, in our algorithm has been implemented with a fixed $\alpha = 0.1$. *Construction* runs until it is verified that, with the remaining unassigned customers, it is not possible to achieve a route with negative reduced cost. Note also that all the negative reduced cost routes generated during the construction phase are kept as solutions of the current PP.

**IteratedLocalSearch.** Once an initial solution (a route) has been built, a local search phase tries to improve it by exploring neighbor solutions. It consists of a Destroy and Repair method based on the one described in Corberán et al. (2019). Here, in the destruction phase of the algorithm, $d$ ($d \in \{1, 3\}$) arcs are removed from the route but always keeping at least one arc in it. Then, in the repair phase, a best improvement strategy was adopted, according to which the customer associated with the minimum profit $\mu_c$ is inserted in the route. The reconstruction phase uses the same stopping criteria as *Construction*. Let $\phi_a$ be the set of arcs in $A_R$ in the current route. *IteratedLocalSearch* stops when $min\{\frac{L}{2}, \frac{|\phi_a|}{2}\}$ iterations without improving the solution are performed.

GRASP is repeated until a maximum computing time (2 seconds) is exceeded or a route including all the arcs in $\bar{A}_R$ has been built. All the negative reduced cost routes generated are kept as solutions of the current PP.

### Restricted master heuristic

In order to speed up the B&P algorithm, and also improve the convergence speed of the column generation, we implemented a *restricted master heuristic* as defined in Joncour et al. (2010). The basic idea behind restricted master heuristics is to solve, by means of a general mixed integer linear programming (MILP) solver, the MP (in our case model (8.4)) defined over a subset of the available columns. We run the

restricted master heuristic every $\Delta$ column generation iterations and whenever an optimal solution for the RLMP has been computed.

The heuristic is immediately terminated when it is triggered and the current (optimal) solution is not feasible. Otherwise, let $\bar{R}$ be the set of the routes associated with the $\bar{\lambda}^{kr}$ variables defining the current (optimal) feasible solution to the RLMP. The routes in $\bar{R}$ are used to initialize an integer program similar to (8.4) to be solved by means of a general MILP solver. In the new integer program, $\boldsymbol{\lambda}$ variables are no more indexed by vehicle. The program considers binary variables $\lambda^r$ assuming value 1 if route $r$ is selected to be assigned to one of the vehicles. According to the new definition of the $\boldsymbol{\lambda}$ variables, coefficients $s_c^r$ and $d^r$ play respectively the role of coefficients $s_c^{kr}$ and $d^{kr}$ in (8.4). The program reads as follows:

$$\bar{W} = \min \quad w \tag{8.8a}$$

$$s.t. \quad \sum_{r \in \bar{R}} s_c^r \lambda^r \geq 1 \qquad \forall c \in \mathbb{H} \tag{8.8b}$$

$$\sum_{r \in \bar{R}} \lambda^r \leq K \tag{8.8c}$$

$$d^r \lambda^r - w \leq 0 \qquad \forall r \in \bar{R} \tag{8.8d}$$

$$\lambda^r \in \{0, 1\} \qquad \forall r \in \bar{R} \tag{8.8e}$$

$$w \geq 0 \tag{8.8f}$$

The new $\boldsymbol{\lambda}$ variables allow to aggregate constraints (8.4b) in (8.4) and formulate them as (8.8c), to disregard constraints (8.4d), and, in general, to avoid symmetries in the solution space. This comes at the expense of an increase in the number of constraints (8.8d) required to define the maximum length.

Whenever an optimal solution to (8.8) is found, a new upper bound $\bar{W}$ (for $w^1$) becomes available. Hence, column generation is restarted to solve the RLMP by considering only vehicle routes with length smaller than, or equal to, $\bar{W} - 1$.

### Overall algorithm overview

The RLMP is initialized by means of set of columns $\overline{\mathcal{C}} \cup \mathcal{C}$. Set $\overline{\mathcal{C}}$ includes a high cost dummy column for each customer $c \in \mathbb{H}$, by means of which constraints (8.4a) are satisfied. In particular, the column for a given customer $c \in \mathbb{H}$ has a coefficient 1 on the row corresponding to the constraint associated with the customer, whereas all the other coefficients of the column are 0. Similarly, a high cost dummy column is further included in $\overline{\mathcal{C}}$ for each vehicle $k \in \mathbb{K}$ to satisfy constraints (8.4b). All dummy columns have null length. At the root node of the branch-and-bound tree, $\mathcal{C}$ is empty.

In any other node of the tree, $\mathcal{C}$ includes the columns that were in the optimal basis of the RLMP at the father node, and that correspond to routes that are feasible with respect to the active branching constraints and the current value of W. Finally, when an heuristic solution of value $\overline{W}$ is available from scratch, at the root node of the tree is inserted a column for each route (defining the solution) whose length is strictly lesser than $\overline{W}$.

Then, iteratively, the RLMP is solved to optimality or proved to be infeasible. First, at each iteration, the RLMP is solved and the dual variable values retrieved. If no dummy column appears in the current basis, all columns in $\overline{\mathcal{C}}$ are extracted from (the constraint matrix of) the RLMP.

Let $\mathcal{PC}$ be the set of columns generated so far during the execution of the whole B&P algorithm (the so called pool of columns). Columns in $\mathcal{PC}$ correspond to routes that are feasible with respect to the current value of W. Before solving the PPs, negative reduced cost columns are searched for among those included in $\mathcal{PC}$. $\mathcal{PC}$ is scanned sequentially and at most K negative reduced cost columns are selected to be inserted into the RLMP. A column in $\mathcal{PC}$ is eligible for selection if (i) the corresponding route is feasible with respect to the active branching constraints, (ii) none of the previous selected columns is associated with the same PP associated with it, and (iii) the corresponding route serves at least one customer not served by any of the routes corresponding to previous selected columns. Let $\overline{\mathcal{PC}}$ be the subset of columns selected and extracted from $\mathcal{PC}$. If $|\overline{\mathcal{PC}}| > 0$, columns in $\overline{\mathcal{PC}}$ are inserted into the RLMP and a new iteration is started, otherwise the PPs are solved in order to eventually find new negative reduced cost columns.

An attempt is made to solve the PPs, as described in Section 8.4.1, by means of the GRASP heuristic (Section 8.4.1). In case no negative reduced cost column is generated, the solution mechanism described in Section 8.4.1 is reapplied by solving the PPs to optimality through the B&C algorithm (Section 8.7). If, again, no negative reduced cost column is found, it means that either the optimal solution of the current RLMP is also optimal for the LMP, or the LMP is infeasible (some dummy columns are in the basis corresponding to the dual variable values for the current iteration). In both cases the column generation algorithm terminates. Otherwise, the negative reduced cost columns generated are inserted into the RLMP and a new iteration is started. When the column generation algorithm ends, all the non-dummy columns defining RLMP are (re)inserted in $\mathcal{PC}$.

Finally, as mentioned in Section 8.4.1, the column generation algorithm takes advantage of a restricted master heuristic. This helps in speeding up its convergence as well as the convergence of the whole B&P algorithm. Actually, the cardinality of the

set $\bigcup_{k\in\mathbb{K}} R^k$ depends on the value of $W$. The lesser the value of $W$, the smaller the cardinality of the set, and this eventually allows to solve faster the PPs with both the GRASP heuristic and the B&C algorithm. In turn, the smaller the cardinality of the set $\bigcup_{k\in\mathbb{K}} R^k$, the greater the dual bound generated by solving to optimality the RLMP, and, in general, tighter dual bounds associated with the nodes of the tree imply a faster convergence of the B&P algorithm. The restricted master heuristic is run every $\Delta$ column generation iterations, before solving the RLMP and retrieving the dual variable values, and whenever an optimal solution for the RLMP has been computed. Each time a new improving feasible solution is found, a new upper bound $\overline{W}$ (for $w^1$) becomes available. The value of $W$ is updated accordingly. All the columns in the RLMP corresponding to routes that do not satisfy constraint (8.7f) with the updated value of $W$ are removed from its constraint matrix. Similarly, columns in $\mathcal{PC}$ corresponding to routes which are no more feasible with respect to the updated value of $W$ are deleted from the set. Columns in $\overline{\mathcal{PC}}$ are reinserted into the RLMP. The solution process of the RLMP is then restarted. In particular, when the improving feasible solution is found after having computed the optimal solution for the current RLMP, the whole column generation algorithm is restarted.

### 8.4.2 Branching rules

Let $(\overline{\boldsymbol{\lambda}}, \overline{\mathbf{w}})$ be the optimal solution of the current RMP. When $(\overline{\boldsymbol{\lambda}}, \overline{\mathbf{w}})$ is fractional, we apply a two-level hierarchical branching scheme. Branching rules are presented in the following in order of priority.

First, we consider an application of the Ryan and Foster's branching rule (see Ryan and Foster (1981)). For each pair of customers $c'$ and $c''$, we define $\alpha_{c'c''} = \sum_{k\in\mathbb{K}} \sum_{r\in R^k} s^{kr}_{c'} s^{kr}_{c''} \lambda^{kr}$ as the sum of the $\lambda^{kr}$ variable values associated with routes that serve both customers $c'$ and $c''$. We select the fractional value $\alpha^*_{c'c''}$ closest to 0.5 such that $0 < \alpha^*_{c'c''} < 1$. On one branch, we set $\alpha^*_{c'c''} = 0$, meaning that the customers $c'$ and $c''$ must be served in different routes (and hence by different vehicles). Whereas, on the other branch, we set $\alpha^*_{c'c''} = 1$, meaning that the two customers have to be served in the same route by the same vehicle. When $\alpha^*_{c'c''}$ is set to 0, the constraint $z^k_{c'} + z^k_{c''} \leq 1$ is inserted in the formulation (8.7) associated with each vehicle $k \in \mathbb{K}$. For the case $\alpha^*_{c'c''} = 1$, the formulation (8.7) associated with each vehicle $k \in \mathbb{K}$ is modified by inserting constraint $z^k_{c'} - z^k_{c''} = 0$. This rule does not introduce symmetries in the solution space, does not alter the structure of the PPs, and, finally, allows the PPs to continue sharing the same feasible region. Thus, as long as only this rule is applied, at each column generation iteration it is possible to

design the sequential solution of the PPs to potentially avoid solving some of them (see Section 8.4.1).

When the solution is fractional and no pair of customers $c'$ and $c''$ exists such that $0 < \alpha^*_{c'c''} < 1$, we branch on the fractional use of an arc by vehicle $k \in \mathbb{K}$. For each $r \in R^k$, let $b^{kr}_{ij}$ be an integer parameter equal to the number of times the vehicle $k$ traverses arc $(i, j)$ while traveling along route $r$. We consider values $\beta^k_{ij} = \sum_{r \in R^k} b^{kr}_{ij} \lambda^{kr}$ and select $\beta^{k^*}_{ij}$ such that $\beta^{k^*}_{ij} - \lfloor \beta^{k^*}_{ij} \rfloor$ is the closest to 0.5. On one branch, the formulation (8.7) associated with vehicle $k$ is modified by considering an upper bound $\lfloor \beta^{k^*}_{ij} \rfloor$ on the use of arc $(i, j)$ and constraint $\sum_{r \in R^k} b^{kr}_{ij} \lambda^{kr} \leq \lfloor \beta^{k^*}_{ij} \rfloor$ is inserted in the LMP. Then, on the other branch, a lower bound $\lfloor \beta^{k^*}_{ij} \rfloor + 1$ on the use of arc $(i, j)$ is considered in the formulation (8.7) for vehicle $k$, and the additional constraint $\sum_{r \in R^k} b^{kr}_{ij} \lambda^{kr} \geq \lfloor \beta^{k^*}_{ij} \rfloor + 1$ is inserted in the LMP. The new constraints inserted in the LMPs of the two branches give rise to additional dual variables, $\gamma^{UB^k}_{ij} \leq 0$ and $\gamma^{LB^k}_{ij} \geq 0$, respectively, that have to be considered in the definition of the reduced cost of the routes in $R^k$. Let $A^{UB^k}$ ($A^{LB^k}$) be the subset of arcs for which dual variables $\gamma^{UB^k}_{ij}$ ($\gamma^{LB^k}_{ij}$) exist. The objective function of the PP associated with vehicle $k$ becomes:

$$\min \quad -\sum_{c \in \mathbb{H}} \mu_c z^k_c - \sum_{(i,j) \in A} d_{ij} x^k_{ij} \rho_k - \theta_k - \sum_{(i,j) \in A^{UB^k}} \gamma^{UB^k}_{ij} x^k_{ij} - \sum_{(i,j) \in A^{LB^k}} \gamma^{LB^k}_{ij} x^k_{ij}$$

This branching rule is sufficient to guarantee the integrality of the solutions. In fact, integer flows on arcs for each vehicle guarantee that the $\boldsymbol{\lambda}$ variables are integer (see Barnhart et al. (1998)). However, the application of this type of rule differentiates the pricing problem feasible regions. Thus, in each node of the subtree arising from their applications, all the PPs have to be eventually solved at each column generation iteration (see Section 8.4.1). This is also the reason why we decided from the beginning to index sets $R^k$ by vehicle index (see Section 8.2.2).

The search tree is explored according to a best-first search strategy.

It is worth mentioning that, even if the optimal solution $(\overline{\boldsymbol{\lambda}}, \overline{\mathbf{w}})$ is fractional, it may happen that at most $K$ distinct routes are selected and fractionally assigned to the different vehicles. Since every $\lambda^{kr}$ variable in the current RMP represents a route $r \in R^k$ which has to be feasible w.r.t. the current upper bound $W$ for $w^1$ (i.e., $d^{kr} < W$), the optimal solution $(\overline{\boldsymbol{\lambda}}, \overline{\mathbf{w}})$ can be converted into an integer feasible solution to (8.4) whose value improves the current upper bound $W$. Whenever this happens, we convert the optimal fractional solution into the corresponding integer solution and update $W$ accordingly.

## 8.5 Primal bound heuristic

In the branch-and-cut and branch-and-price algorithms described in the previous sections, we use the upper bounds provided by the following heuristic, which is based on the multi-start iterated local search matheuristic for the DC-CEARP described in Corberán et al. (2019).

In Corberán et al. (2019), three different criteria for initializing the routes are proposed: *random initialization*, *random selection among the best applicants*, and *weighted selection among the best applicants*. The *random initialization* criterion chooses the first customer of each route completely at random, *weighted selection among the best applicants* chooses the first customer randomly among a set of customers closest to the depot, and *weighted selection among the best applicants* assigns weights to the customers according to their distance to the depot and chooses the initial customer for each route with probability proportional to these weights. We generate one solution with each different initialization criterion.

Since the goal in the MM-CEARP is to minimize the length of the largest route, and there is no maximum length for the routes, we complete the routes by following the *parallel completion* strategy described in Corberán et al. (2019) with some modifications. Let $k_0$ be the longest route among those partially constructed. For all the arcs serving customers that have not been assigned yet, we calculate their insertion cost in all the possible positions of all the routes except $k_0$, and insert the arc according to the cheapest insertion. The customers served by this arc are marked as served. These steps are repeated until all the customers have been assigned to a route.

The constructed solutions are used as initial solutions of an Iterated Local Search (ILS) heuristic. The local search operators used are the exchange of pairs of arcs belonging to two different routes the routes (*2-Exchange*), and the reconstruction of partial solutions, *Destroy-Repair*. Both are well-known perturbation operators, and their implementation details can be found in Corberán et al. (2019).

New solutions are iteratively generated and improved using the above operators until a time limit $t_l$ is reached or a certain number of iterations $i_{max}$ are performed without improving the best solution. After some a preliminary tests, we decided to set the maximum number of iterations ($i_{max} = 10$) and the time limit ($t_l = 100$) in order to balance the amount of time and the quality of the solution. Algorithm 4 shows the structure of the proposed heuristic.

**Input:** $G$, $\mathbb{H}$, $i_{max}$, $t_l$
**Output:** $S_{best}$
$i \leftarrow 0$;
$S_{best} \leftarrow \emptyset$;
**while** $t_l$ is not reached AND $i \leq i_{max}$ **do**
  **for** each Constructive algorithm **do**
    $S_c \leftarrow$ Constructive algorithm();
    $S_l \leftarrow \mathsf{ILS}(S_c)$;
    **if** $S_l$ is better than $S_{best}$ **then**
      $S_{best} \leftarrow S_l$;
      $i \leftarrow 0$;
  $i \leftarrow i + 1$;

**Algorithm 4:** Overall heuristic algorithm for the MM-CEARP

## 8.6 Computational experiments

In this section we test the performance of the two exact algorithms proposed in this chapter. We have conducted all the computational experiments on a desktop PC with an Intel(R) Core(TM) i7 clocked at 3.4GHz CPU, with 32 GBytes of RAM and running Windows 10 Enterprise 64 bits. The branch-and-cut algorithms described in Sections 8.3 and 8.7 have been implemented in C++ using IBM ILOG CPLEX 12.10 with Concert Technology, and compiled in release mode with MS Visual Studio Community 2015. Also the B&P algorithm have been implemented in C++ and compiled in release mode with MS Visual Studio Community 2015. In particular, the callable library of CPLEX 12.10 was used for (re-)optimizing the RLMPs. Finally, all the algorithms have been compiled by allowing a single thread of execution.

The experiments were carried out with a CPU time limit of two hours. For the B&C algorithm described in Section 8.3, we turned off CPLEX heuristic algorithms and activated CPLEX own cuts (including zero-half cuts) in automatic mode. We fixed to zero the tolerance of the optimality gap and selected the best bound branching strategy.

The instances used and the computational results obtained are described in what follows.

### 8.6.1 Instances

In Ávila et al. (2017), four different data sets for the Distance-Constrained CEARP (DC-CEARP) were proposed. Two of them were based on the street networks of two Spanish towns, *Albaida* and *Madrigueras*, and the other were based random graphs with 50 and 75 vertices, *Random50* and *Random75* respectively. In total, 72 instances

were defined. Moreover, by considering the number of vehicles allowed to serve the customers, ranging between 2 and 5, the total number of instances addressed was 251.

Contrary to what happens in the DC-CEARP, where there is a maximum distance for each route that determines the minimum number of vehicles needed, in the Min-Max CEARP there is no such limitation. Thus, the 72 DC-CEARP instances can be solved for any number of vehicles. However, depending on the characteristics of the instance, it may not make sense to use a very high number of vehicles. It may happen that a customer (or a set of customers) is very far from the depot, so the length of the longest route can be determined by the trip to serve this customer and in this case the optimal objective value will not decrease if we increase the number of vehicles. To address this issue, given an instance, we compute for each customer the length of the shortest route traveling from the depot to an arc of the customer, serving it and going back to the depot. Then, the longest of these routes provides a trivial lower bound for the MM-CEARP associated with the instance. Each instance is solved iteratively with $k = 2, 3, ...$ vehicles. If for a given value of $k$, the optimal solution cost is equal to the trivial lower bound, we set $k - 1$ as the maximum number of vehicles for this instance. A total of 258 instances have been defined with a minimum of 2 vehicles and a maximum of 11.

The characteristics of these instances, grouped by sets, are shown in Table 8.1. The number of instances per set is given in column *#Inst*, and the maximum number of vehicles for which the instances in this set have been solved in column *Max K*. The remaining columns report the minimum and maximum number of arcs, arcs in $A_R = H_1 \cup \cdots \cup H_L$, arcs in $A_{NR} = A \setminus A_R$, and customers, respectively, for the instances in each set.

Table 8.2 summarizes the distribution of the instances according to the number of vehicles. Since the number of instances with more than 5 vehicles is very limited, in the analysis of the computational results outlined in the next section we grouped the 44 instances with 6 or more vehicles and denoted the group as M6.

All these instances, as well as their best known solutions, can be downloaded from *http://www.uv.es/corberan/instancias.htm*.

### 8.6.2 Computational Results

This section presents the results of our computational experiments to compare the performance of the B&C and B&P algorithms. Table 8.3 reports the results on the 258 test instances grouped by dataset. The first column indicates the group of instances, while the second column shows the number of instances in each group. The remaining

| | | | | $\|A\|$ | | $\|A_R\|$ | | $\|A_{NR}\|$ | | $\|\mathbb{H}\|$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | #Inst | Max $K$ | $\|V\|$ | Min | Max | Min | Max | Min | Max | Min | Max |
| *Albaida* | 81 | 8 | 116 | 259 | 305 | 124 | 172 | 109 | 162 | 18 | 33 |
| *Madrigueras* | 105 | 11 | 196 | 453 | 544 | 224 | 305 | 197 | 281 | 22 | 47 |
| *Random50* | 24 | 5 | 50 | 296 | 300 | 105 | 292 | 7 | 193 | 10 | 100 |
| *Random75* | 48 | 8 | 75 | 448 | 450 | 143 | 438 | 10 | 305 | 15 | 150 |

Table 8.1: Characteristics of the MM-CEARP instances

| #Veh | *Albaida* | *Madrigueras* | *Random50* | *Random75* | TOTAL |
|---|---|---|---|---|---|
| *2* | 24 | 24 | 11 | 12 | **71** |
| *3* | 21 | 22 | 9 | 12 | **64** |
| *4* | 17 | 17 | 3 | 9 | **46** |
| *5* | 9 | 16 | 1 | 7 | **33** |
| *6* | 6 | 12 | - | 4 | **22** |
| *7* | 3 | 7 | - | 3 | **13** |
| *8* | 1 | 4 | - | 1 | **6** |
| *9* | - | 1 | - | - | **1** |
| *10* | - | 1 | - | - | **1** |
| *11* | - | 1 | - | - | **1** |
| | **81** | **105** | **24** | **48** | **258** |

Table 8.2: Number of instances grouped by dataset and number of vehicles

columns report, for each algorithm, the number of optimal solutions found (# opt) and the average CPU time in seconds (Time). The last row summarizes the results for all the instances.

| | | B&C | | B&P | |
|---|---|---|---|---|---|
| | # Inst | # opt | Time (sec) | # opt | Time (sec) |
| *Albaida* | 81 | 61 | 2025.3 | 74 | 750.1 |
| *Madrigueras* | 105 | 27 | 5643.4 | 39 | 4806.7 |
| *Random50* | 24 | 18 | 1911.9 | 21 | 1394.7 |
| *Random75* | 48 | 19 | 4513.8 | 23 | 4085.9 |
| | **258** | **125** | **3950.2** | **157** | **3081.6** |

Table 8.3: Summary of the results on the 258 instances.

It can be seen in Table 8.3 that the B&C algorithm reaches the optimal solution in 48.45% of the instances with 3950.2 seconds on average. Meanwhile, the B&P algorithm is able to solve 60.85% of the instances to optimality in 3081.6 seconds on average. Looking at the *Albaida* set, the B&P algorithm reaches the optimal solution in over 91% of the instances, while less than 40% of the instances have been solved optimally in the *Madrigueras* set. As for the randomly generated instances, the B&P algorithm solves 21 out of 24 of the *Random50* instances, and 23 out of 48 of the *Random75* ones. Although the number of optimal solutions obtained is lower in the case of the B&C algorithm (between 14% and 30% lower, depending on the set), the conclusions are similar when comparing its behavior for the different sets.

| | | B&C | | B&P | |
|---|---|---|---|---|---|
| | # Inst | # opt | Time | # opt | Time |
| *2* | 71 | 66 | 839.8 | 53 | 2115.7 |
| *3* | 64 | 35 | 3487.9 | 43 | 2625.8 |
| *4* | 46 | 16 | 4993.7 | 25 | 3453.5 |
| *5* | 33 | 5 | 6335.1 | 15 | 3977.8 |
| *M6* | 44 | 3 | 6762.3 | 21 | 4242.4 |
| | **258** | **125** | **3950.2** | **157** | **3081.6** |

Table 8.4: Computational results for all the instances grouped by number of vehicles

In Table 8.4 we can see the results for the instances grouped according to the number of vehicles. The B&C algorithm achieves the best results for the instances with 2 vehicles, being able to solve 66 out of 71 instances in short computing times. However, the performance of this algorithm degrades when the number of vehicles increases. This does not happen for the B&P algorithm, which shows a more robust behaviour and outperforms the B&C algorithm for the instances with 3 or more vehicles. This can be clearly seen in the Figure 8.2, which shows for the B&C and B&P algorithms the variation of the number and percentage of optima as a function of the number of vehicles.

In order to study the behavior of the algorithms w.r.t. computing time, we use the performance profiles described by Dolan and More (2002). Let $\mathcal{S}$ be the set of algorithms and $\mathcal{P}$ the set of instances. Let $t_{p,s}$ be the computing time required by algorithm $s \in \mathcal{S}$ to solve instance $p \in \mathcal{P}$. The *performance ratio* is then defined as $r_{p,s} = t_{p,s}/\min\{t_{p,s} : s \in \mathcal{S}\}$. If algorithm $s$ is not able to solve the instance $p$ within the time limit, we set $r_{p,s} = \infty$. Then, the *performance profile* of each algorithm is defined as

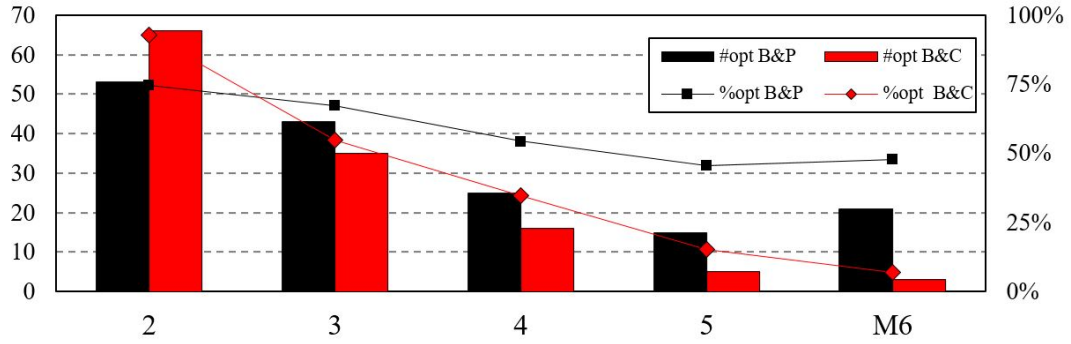


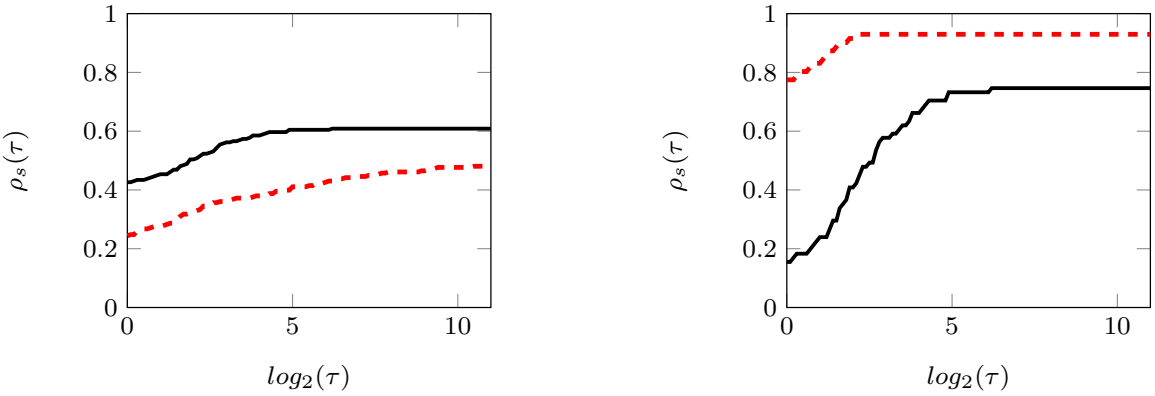

Figure 8.2: Instances per vehicle solved optimally.

$$\rho_s(\tau) = \frac{|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}|}{|\mathcal{P}|},$$

and represents the percentage of instances that can be solved by $s$ within a factor $\tau$ of time with respect to the fastest algorithm. Note that, since the computing times are very different, we plot the results on a logarithmic scale. Note that $\rho_s(0)$ is the percentage of optimally solved instances for which algorithm $s$ is the fastest. Figure 8.3 depicts the performance profiles of the B&C (red dotted line) and the B&P (solid black line) algorithms overall and for the instances grouped by number of vehicles.

Comparing the performance profile for all the instances (Figure 8.3a) in $\tau = 0$, we can see that the B&P algorithm is the fastest in 42.6% of the instances, while the B&C algorithm only in 24.41%. It is interesting how the curve for the B&P algorithm increases rapidly in the interval $[2, 4]$, reaching almost its maximum around 60%. This means that the B&P algorithm solves another 15% of instances using between 4 and 16 times the computing time used by the B&C algorithm. Conversely, the curve of the B&C algorithm shows that this last is slower at reaching optimal solutions. At $log_2(\tau) = 11$ we have already reached the maximum number of optimal solutions for both algorithms.

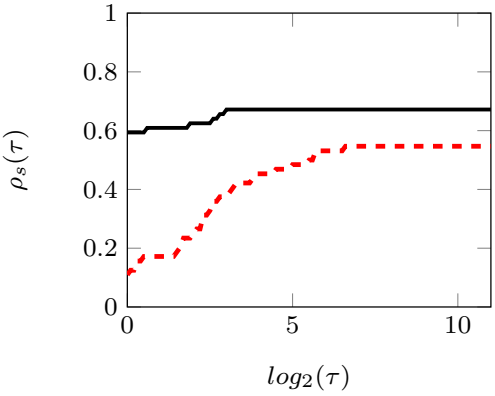
Figure 8.3b shows the performance profile of both algorithms for the instances with 2 vehicles. The B&C algorithm, as expected, is faster on this subset of instances, getting the shortest time in 77.46% of the cases, against a corresponding percentage of only 15.49% associated with the B&P algorithm. However, the results are completely different at the increase of the number of vehicles. From Figure 8.3c we can see that B&P algorithm becomes the fastest in almost 60% of the instances while the B&C algorithm only in just over 10%. Note that in this case, despite taking more time, the B&C algorithm is able to find almost 55% of the optima. Figures 8.3d, 8.3e and 8.3f



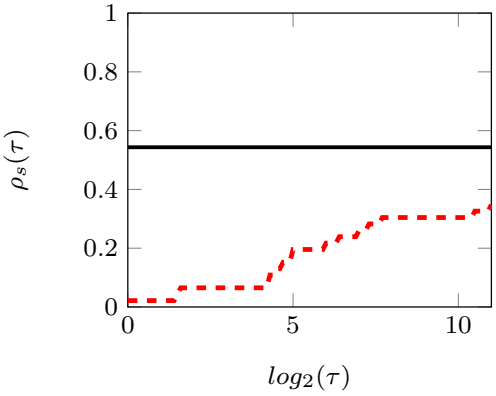

(a) Performance profile - Overall



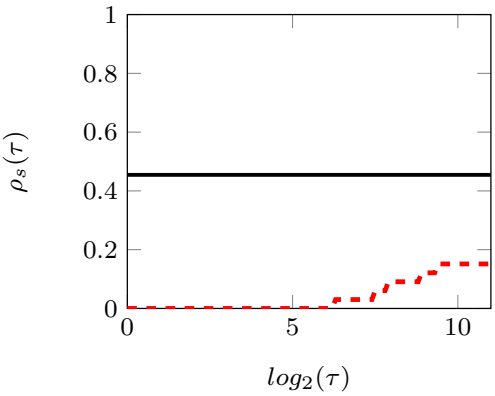

(b) Performance profile - 2 vehicles



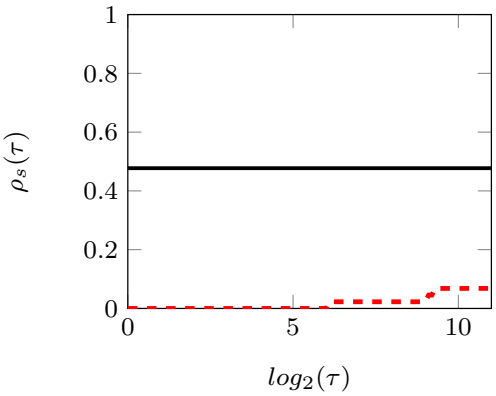

(c) Performance profile - 3 vehicles




(d) Performance profile - 4 vehicles




(e) Performance profile - 5 vehicles




(f) Performance profile - M6 vehicles


Figure 8.3: Performance profiles for different number of vehicles.

show the performance profiles for 4, 5, and 6 or more vehicles, respectively. It can be seen that the B&P algorithm completely dominates the comparison since, despite being below 60% in number of optima, it is the fastest algorithm in practically all

the instances where the optimum is found, while the B&C algorithm needs a lot more time to be able to obtain fewer optimal solutions (at most 3 optima for instances with 6 or more vehicles).

| | Instances with LB | | | | | | | Instances without LB | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | B&C | | | B&P | | | | B&C | | |
| | # Inst | # opt | Time | Gap(%) | # opt | Time | Gap(%) | # Inst | # opt | Time | Gap(%) |
| *2* | 59 | 59 | 245.0 | 0.00 | 53 | 1081.6 | 0.38 | 12 | 7 | 3764.1 | 4.09 |
| *3* | 57 | 35 | 3032.0 | 1.47 | 43 | 2064.0 | 1.19 | 7 | 0 | 7200.0 | 11.67 |
| *4* | 41 | 16 | 4724.6 | 5.43 | 25 | 2996.6 | 1.93 | 5 | 0 | 7200.0 | 22.54 |
| *5* | 31 | 5 | 6279.3 | 8.32 | 15 | 3770.0 | 3.80 | 2 | 0 | 7200.0 | 25.31 |
| *M6* | 42 | 3 | 6741.5 | 7.68 | 21 | 4101.6 | 2.78 | 2 | 0 | 7200.0 | 19.72 |
| | **230** | **118** | **3733.9** | **3.86** | **157** | **2580.2** | **1.76** | **28** | **7** | **5727.5** | **10.89** |

Table 8.5: Gap comparison on the instances with and without LB computed by the B&P algorithm

It is interesting to point out that, for some instances, the B&P algorithm was not able to terminate solving the linear master problem at the root node, so no lower bound has been computed by the algorithm for these instances. In Table 8.5, we first study the gaps between the lower and upper bounds computed by each algorithm for those instances for which the B&P algorithm was able to find a lower bound. If $LB$ and $UB$ denote the lower and upper bounds found by a given algorithm in an instance, the gap is computed as $\frac{UB-LB}{UB} * 100$. The instances are grouped according to the number of vehicles. For each group, the second column gives the number of instances with a lower bound provided by both algorithms, columns three to five, and six to eight, report the number of optima found, the average computing time, and the average gap obtained by the B&C and B&P algorithm, respectively. Furthermore, the last four columns in Table 8.5 provide the results obtained with the B&C algorithm for the instances where no lower bound was computed by the B&P algorithm in two hours of computing time. In particular, for each number of vehicles, columns 9-12 give the number of such instances, the number of those optimally solved, the average computing time, and the average gap obtained with the B&C algorithm.

Again, we can see that the B&C algorithm is effective on the instances with 2 vehicles, where all the 59 instances with LB in B&P, and 7 out of 12 more instances, are optimally solved. Furthermore, the average gap for the 5 unsolved instances, 4.09%, is quite small. The effectiveness of the B&C algorithm decreases for the instances with 3 vehicles, and it is inferior to that of the B&P algorithm. Nevertheless, the B&C algorithm is able to compute lower bounds, associated with an average gap of 11.67%, for 7 instances for which the B&P algorithm can not. For the instances with more vehicles, the performance of the B&P algorithm is clearly superior to that of

the B&C algorithm. Note that with respect to the B&C algorithm, both the number of unsolved instances and the average gap increase steadily as the number of vehicles increases, while this behaviour is not so pronounced for the B&P algorithm, whose reported average gaps do never exceed 3.80%. Moreover, the number of instances in which no lower bound is computed by the B&P algorithm decreases as the number of vehicles grows. It is also worth noting that the gaps reported for the B&C algorithm with respect to these instances are very high, which seems to indicate that they are particularly difficult.

## 8.7 Conclusions

This chapter addresses the Min-Max Close-Enough Arc Routing Problem, where a fleet of homogeneous vehicles has to serve a set of customers in such a way that the lengths of their routes are balanced. For this problem we have proposed two different models. The first one is an arc-based formulation, with arc and servicing variables, that has been used to develop a branch-and-cut algorithm, while the second one is a route-based set-covering formulation used to design a branch-and-price algorithm in which the pricing problems are solved by means of a branch-and-cut algorithm. Moreover, a heuristic to provide the exact algorithms with initial feasible solutions has been implemented. An extended computational analysis has been carried out, where we have studied the performance of the algorithms on 258 instances with up to 196 vertices, 544 arcs, 150 customers, and 11 vehicles. The results show that the branch-and-cut algorithm achieves the best results for the instances with 2 vehicles, while the performance of the branch-and-price algorithm is better for the instances with 3 or more vehicles. Overall, we have been able to optimally solve 173 out of these instances in two hours of computing time.

# Chapter 9

# Conclusions and future work

In this thesis we have studied three NP-*hard* combinatorial optimization problems that arise in the context of close-enough arc routing problems. The first is the Profitable CEARP for a single vehicle, and the second and third are the Distance-Constrained CEARP and the Min-Max CEARP, respectively, both for multiple vehicles. The aim of these problems is to model specific situations in which customer service does not necessarily have to be done on the site, but it is sufficient to get close enough to it. An in-depth study of the problems has been carried out by addressing specific objectives and/or constraints and we have developed exact and heuristic methods to solve them. This chapter summarizes the main contributions of the thesis in Section 9.1 and the future lines of research in Section 9.2.

## 9.1 Contributions

First of all, Chapters 2, 3, and 4 present the context in which the problems studied in this thesis arise. Basic concepts of Mathematical Programming to non-expert readers are provided in Chapter 2, where several definitions and results of graph theory, linear and integer programming, and polyhedral theory and combinatorics are given. Chapter 3 provides an overview of some classic routing problems that have been studied in depth in the literature. To conclude the introductory section, Chapter 4 presents the state of the art and some real-world applications of close-enough routing problems. This chapter includes an overview of the CETSP along with a more detailed study of all the existing research carried out so far for CEARP, the original ARP to which the problems studied in this thesis generalize. The CEARP has been described in the literature as an innovative arc routing problem in which the customers are not necessarily arcs or edges of a network, but the associated service can be performed

from any arc or edge traversed that is close enough to the customer. It consists of finding a minimum cost tour servicing all the customers.

Subsequently, in Chapter 5 we study in depth the Profitable CEARP, in which a profit is associated with each customer and is collected (only once) when the customer is serviced. The goal is to find a tour that maximizes the difference between the total profit collected and the travel distance. The chapter begins by formally defining the problem, proposing a formulation, and conducting the polyhedral study. We have shown that some inequalities of the formulation always define facet and others do so under specific conditions. In addition, from the properties of the PCEARP tours, other valid inequalities not obtained directly from the formulation are presented here, which strengthen the description of the polyhedron. To solve the Profitable CEARP, a heuristic and a branch-and-cut algorithm have been designed and implemented. The heuristic combines a constructive procedure and a local search, and it provides the exact algorithm with initial lower bounds. In the branch-and-cut algorithm, all separation procedures for the identification of violated inequalities and the order in which they are applied have been studied. Both algorithms have been adjusted and evaluated through extensive experiments and statistical analysis, and to test them, four different sets of instances specific to this problem were generated with up to 800 customers, 400 vertices and 2000 arcs. According to the results of the computational experiments, this exact procedure has been able to optimally solve large instances with up to 600 customers, 300 vertices, and 1500 arcs, in less than one hour.

Chapters 6 and 7 deal with the Distance-Constrained CEARP. It consists of finding a set of routes leaving from and entering at the depot and serving all the customers, such that the length (in distance or time) of each route does not exceed a certain value. The objective is to minimize the total length traversed.

Chapter 6 addresses the Distance-Constrained CEARP, formally defining the problem, introducing the notation used, and presenting the most promising formulation proposed in Ávila et al. (2017). Given that the values of the maximum distances per route are very tight and make it difficult to solve the Distance-Constrained CEARP, there are a good number of unsolved instances in the previous work of Ávila et al. Therefore we have proposed a multi-start matheuristic that incorporates an effective branch-and-cut method for the CEARP in order to optimize the routes obtained. In the computational experiments, we present the results obtained with two versions of the algorithm, considering a maximum number of iterations and a time limit, respectively. In general, the proposed approach finds feasible solutions for almost all the instances and optimal solutions for more than 70% of them.

Chapter 7 also covers the Distance-Constrained CEARP, but in this case a more in-depth study is carried out. We start by proposing a new formulation that combines the best features of the previously existing ones since, despite having more variables, we want to strengthen its linear relaxation. For this formulation we have carried out an exhaustive study of its associated polyhedron and we have proposed several families of valid inequalities. Furthermore, based on the separation algorithms for the new inequalities, we have proposed a branch-and-cut algorithm that provides very good results. Extensive computational experiments have been performed on a set of benchmark instances to analyze the contribution of the valid inequalities and the separation algorithms presented. The gaps in the root node and the performance profiles of the different versions of our branch-and-cut procedure (using different combinations of separation algorithms) are compared. The results of the two best versions of our branch-and-cut algorithm are compared with those obtained with the matheuristic and the exact methods presented in Corberán et al. (2019) and Ávila et al. (2017), respectively. The best version of our branch-and-cut algorithm is capable of optimally solving instances with up to 140 customers, 196 vertices, 544 arcs, and 5 vehicles to in two hours of computing time.

Chapter 8 deals with the Min-Max CEARP for a fleet of homogeneous vehicles. Taking into account the type of applications of the CEARP, we have studied here the version that tries to balance the length of the routes by minimizing the duration of the longest route. The problem consists of finding a set of routes, all of them starting and ending at the depot, jointly servicing all customers, and with a minmax objective. We begin by presenting and providing two different models for the problem: an arc-based formulation, with arc and service variables, that has been used to develop a branch-and-cut algorithm, and a route-based set-covering formulation, which has been used for the development of a branch-and-price algorithm. We also describe a heuristic algorithm used to provide initial feasible solutions to exact algorithms. The branch and cut is based on the method presented in Ávila et al. (2017). In the branch and price we have implemented two atypical techniques of these algorithms when addressing routing problems, the first-level rule applied in the branching scheme (based on Ryan and Foster's branching rule) and a branch-and-cut algorithm to solve the pricing problems to optimality instead of using dynamic programming. The first one allows to recover integer solutions at the expense of a diversification of the sets of feasible routes, does not introduce symmetries in the solution space, does not alter the structure of the pricing problems, and, finally, allows pricing problems to continue sharing the same feasible region. We have performed an extended computational analysis in which we have compared the performance of the exact algorithms in four sets of instances tested with a minimum of two vehicles and a maximum of 11. The results

show that the branch-and-cut algorithm achieves the best results for the instances with two vehicles, while the branch and price performs best for instances with three or more vehicles.

## 9.2 Future lines of research

Although the thesis concludes here, research is an endless field. The research carried out in this work can continue to address other constraints and variants of the problems. In fact, many of the ideas on which some of the algorithms designed are based could be directly or easily adapted in other related arc routing problems, as well as many of the ideas presented in the theoretical parts could also be used in other related problems.

Given the excellent results of the branch-and-price algorithm in solving Min-Max CEARP instances with a large number of vehicles, it would be worth developing an algorithm of the same type for the Distance-Constrained CEARP. It has been studied in the literature that branch-and-cut algorithms are very efficient in the solution of CEARP, and here we show that they are very efficient in solving the multi-vehicle variant when the size of the fleet is small but, when the number of vehicles increases, a branch-and-price algorithm is much more appropriate.

With respect to the Profitable Close-Enough Arc Routing Problem considered in Chapter 5, it could be extended in several ways by progressively adding characteristics of related real problems. For example, the profit per customer could be defined by the time it takes to perform the service. Also, if a single vehicle is not capable of performing all the services, a fleet of vehicles (or multiple routes for a single vehicle) is required. It could be considered that all vehicles have the same characteristics or that we have a heterogeneous fleet. We could also set a maximum distance/time for each route, or minimize the cost of the longest route.

Except for the matheuristic proposed for the DC-CEARP, throughout this thesis we have approached the problems from a theoretical point of view, the results of which have been used to develop exact algorithms for the optimal solution of the problems. All the problems studied here are combinatorial problems that are NP-*hard*, so there are no algorithms that find the optimal solutions in polynomial time. Thus, it may happen that in certain circumstances it is necessary to have good quality solutions for real problems in short computing times. In this sense, we think that, as a result of the knowledge acquired during the study of the problems dealt with in this thesis, heuristic algorithms could be designed and implemented for the three problems studied that improve the quality and time of the existing ones.

# Appendix A

# Resumen Extendido

A pesar de carecer de datos específicos, se estima que el sector del transporte representa aproximadamente el 64% del consumo mundial de combustible, el 27% del consumo total de energía y el 23% de las emisiones mundiales de dióxido de carbono ($CO_2$) relacionadas con la energía. Además, se prevé que el impacto medioambiental del sector del transporte aumente de forma drástica en los próximos años debido al efecto de la globalización, que ha eliminado barreras haciendo posible la accesibilidad a todos los lugares, productos y servicios del mundo. Por ello, el transporte se sitúa como uno de los principales retos en materia de desarrollo, para impulsar la prosperidad y lograr así un entorno sostenible que reduzca el consumo energético. Con este fin, la tecnología desarrollada para el análisis y la explotación de la enorme cantidad de datos -que están generando fuentes como la sensorización, los contenidos de Internet o el comportamiento de los usuarios- está ayudando a definir los patrones y las necesidades de circulación, mejorando de este modo la calidad y la eficiencia de las soluciones de transporte. Al mismo tiempo, el acceso a las nuevas tecnologías está alterando el comportamiento y las expectativas de los consumidores, promoviendo una creciente tendencia a la inmediatez en nuestra sociedad.

El sistema de transporte siempre ha sido un factor crucial para las empresas que transportan mercancías o prestan servicios y hoy en día se ha convertido en un elemento diferencial en sus esfuerzos por maximizar la satisfacción del cliente. Por ello, el transporte no se limita a trasladar mercancías de un lugar a otro o a prestar un servicio concreto, sino que es un proceso estratégico que busca reducir los costes logísticos y mejorar la experiencia del cliente/consumidor. En la actualidad, es prioritario que las empresas adopten una calidad de servicio óptima que proporcione una mayor eficiencia en los tiempos y una mayor adaptabilidad, seguridad y resiliencia.

La logística del transporte puede optimizarse mediante una planificación adecuada de las rutas con el fin de maximizar la productividad, ahorrar en costes de transporte y aumentar la rentabilidad. Se gastan cantidades exorbitantes de dinero en combustible, funcionamiento y mantenimiento de vehículos, así como en mano de obra. Por ello, después de años de conocer su función e importancia, cada vez somos más conscientes de la importancia de optimizar la logística y convertirla en una ventaja competitiva y de que esta optimización de la logística es una cuestión clave a la hora de reducir gastos. Se estima que el uso de procedimientos informatizados para la planificación del proceso de distribución produce ahorros sustanciales (generalmente de entre el 5% y el 20%) en los costes globales de transporte. De hecho, el proceso de transporte implica a todas las etapas de los sistemas de producción y distribución y representa generalmente del 10% al 20% del coste final de las mercancías. En este sentido, una pequeña mejora en los problemas de rutas puede suponer un enorme ahorro logístico en términos absolutos.

En el mundo académico, la optimización de rutas consiste en determinar la ruta más rentable teniendo en cuenta diversos factores, como las limitaciones de los vehículos, los controles de costes, las ventanas de tiempo para el servicio, las limitaciones de recursos en el proceso de carga en el depósito, etc. Así, los problemas de rutas han sido definidos como el diseño de rutas óptimas desde un depósito hasta un conjunto de destinos para las cuales existen restricciones específicas. Estos problemas han atraído la atención de muchos investigadores y profesionales durante los últimos 60 años debido a los retos matemáticos que conlleva su estudio y resolución, y también debido a la motivación que supone el gran impacto económico de las mejoras encontradas. Se diferencian muchos tipos de problemas de rutas según el tipo de mercancía o servicio a realizar, las características de la flota de vehículos (tamaño, capacidad, autonomía), la distancia y el nivel de servicio de los clientes, etc.

La mayoría de los problemas de rutas son extremadamente difíciles de resolver de forma óptima en la práctica, por lo que se clasifican en función de las especificaciones de las situaciones reales a modelar. Una clasificación general de estos problemas de rutas se hace dependiendo de si los clientes se representan mediante nodos en un grafo (problemas de rutas de nodos, NRP), y aquellos en los que el servicio se realiza en los arcos o aristas (problemas de rutas de arcos, ARP). Aunque tradicionalmente la investigación en problemas de rutas se ha centrado más en los problemas de rutas de nodos, la literatura sobre problemas de rutas por arcos está creciendo cada día, y el número y la eficiencia de los algoritmos diseñados para estos problemas han aumentado considerablemente en los últimos años. En esta tesis, nos centraremos en el marco de los problemas de rutas por arcos o ARPs.

El ARP se define como el diseño de una ruta (o varias rutas) de tal manera que todos los arcos y/o aristas de un grafo que requieren ser servidos deben ser recorridos. Si nos remitimos al origen, el primer problema de rutas por arcos fue el Problema del Cartero Chino (CPP), introducido por el matemático chino Guan en el año 1962. El nombre del problema se debe a que el autor planteó la situación en la que se encontraba un cartero que quería encontrar un camino de longitud mínima que le permitiera repartir todo el correo. Dado un grafo, el CPP pretende encontrar un camino cerrado de coste mínimo que atraviese todos los arcos y/o aristas al menos una vez. Posteriormente, se propuso el Problema del Cartero Rural (RPP), una generalización del CPP, en la que los servicios no tenían que realizarse en todas las calles, sino en un subconjunto de ellas. El objetivo de éste es también encontrar un camino cerrado de mínimo coste que recorra al menos una vez todos los arcos y/o aristas que requieren ser atendidos, pudiendo pasar por el resto para cerrar el camino. Desde entonces, en la literatura se han estudiado de forma detallada un gran número de variantes de estos problemas. Además del reparto de correo, los problemas de rutas por arcos han sido estudiados en la literatura gracias a las muchas aplicaciones que tienen en la organización de tareas, como la recogida de basuras, el servicio de limpieza de nieve, el reparto de leche, la inspección de sistemas de distribución (redes eléctricas, telefónicas o ferroviarias), la limpieza y el riego de calles, etc. Cada año se gastan millones de euros en estas operaciones y el ahorro que se puede conseguir optimizándolas es enorme. El principal reto de estos problemas es que no pueden modelizarse como simples ARPs, sino que cada problema tiene sus propias características. Por lo tanto, la metodología utilizada para resolverlos debe ser específica y debe considerar el contexto específico de cada problema.

Al igual que en otros sectores, las innovaciones tecnológicas relevantes, como los nuevos tipos de dispositivos, la tecnología de identificación por radiofrecuencia (RFID) y la disponibilidad de datos en tiempo real a través de la geolocalización, los flujos de tráfico o la comunicación entre clientes y conductores, han provocado numerosos cambios en el negocio de la logística del transporte. Simultáneamente, la investigación operativa sigue evolucionando y plantea la necesidad de definir y estudiar nuevos problemas, así como la incorporación de nuevas características a los ya existentes. En los últimos años, el desarrollo de las nuevas tecnologías trajo consigo escenarios que no requieren que el vehículo alcance la ubicación del cliente, sino sólo que se acerque a éste para realizar el servicio. Esta situación es conocida como "close-enough" en los problemas de rutas, ya que el vehículo sólo necesita pasar lo suficientemente cerca de la posición del cliente (Close-Enough Routing Problems - CERPs). Si además para cada cliente el servicio se lleva a cabo cuando el vehículo atraviesa los arcos de un grafo, resulta el Close-Enough Arc Routing Problem (CEARP). El CEARP

consiste en encontrar una ruta de coste mínimo que empiece y termine en el depósito y que atraviese algunas de las calles de una red de carreteras de manera que se preste servicio a todos los clientes. Notar que, como un cliente es atendido cuando el vehículo se acerca a una determinada distancia, se conocen qué calles debe recorrer el vehículo para atender a cada cliente. La principal característica de este problema es que, a diferencia de los Problemas de Rutas por Arcos tradicionales, el conjunto de calles que hay que recorrer no se conoce de antemano, sino que es una variable de decisión.

Una de las aplicaciones más directas del CEARP se encuentra en la lectura automática de contadores, ya que la identificación por radiofrecuencia (RFID) permite recoger a distancia los datos de consumo de los contadores de gas, electricidad o agua, en lugar de tener que hacerlo puerta a puerta como era habitual hace años. El contador envía una señal que describe el consumo y que es captada por el receptor si se encuentra a una distancia determinada. Así, dada una red de carreteras en la que se encuentran los contadores, existen vehículos con un receptor de radiofrecuencia que pueden leer el consumo con sólo acercarse a cada contador. En este caso, el vehículo/operador sólo tiene que entrar en la zona de cobertura del contador para realizar el servicio, sin necesidad de visitar físicamente a todos los clientes, lo que supone un ahorro de tiempo y dinero.

Otra aplicación de estos problemas se encuentra en la gestión de inventarios en las grandes empresas, especialmente en aquellas en las que hay muchos productos y, por tanto, comprobarlos uno a uno requiere mucho tiempo. Las etiquetas RFID han revolucionado la gestión de la cadena de suministro al permitir a los responsables de los almacenes registrar el inventario de forma mucho más eficiente de lo que podían hacerlo leyendo los números de las cajas y registrándolos manualmente. Los investigadores del MIT han desarrollado un sistema que permite a los drones aéreos leer las etiquetas RFID desde decenas de metros de distancia e identificar la ubicación de las etiquetas con un error medio de unos 19 centímetros. Por lo tanto, para realizar el inventario, el dron no necesita atravesar todos los pasillos del almacén para la recogida de datos. Los investigadores prevén que el sistema podría utilizarse en grandes almacenes tanto para la supervisión continua, con el fin de evitar desajustes en el inventario, como para localizar artículos individuales, de forma que los empleados puedan responder de forma rápida y fiable a las peticiones de los clientes.

Por otra parte, el CEARP también ha sido utilizado para modelizar el recorrido a realizar por los drones con receptores RFID o cámaras incorporadas que realizan determinadas tareas como el control de calidad de mantenimiento o la vigilancia de las redes. En esta última tarea, los drones no tienen que sobrevolar los puntos o líneas a vigilar, sino sólo acercarse al objetivo a una determinada distancia. Finalmente, en

una red de sensores inalámbricos, donde los sensores están geográficamente distantes entre sí, puede no ser práctico requerir que los sensores se coordinen directamente entre sí para formar una red de comunicación, debido a la restricción de energía. Una posible solución es emplear un robot móvil, que pueda desplazarse a todos los sensores para descargar los datos y finalmente regresar a su estación base (posición de partida).

En esta tesis estudiamos tres problemas de optimización combinatoria NP-*hard* que surgen en el contexto de los Close-Enough ARPs. El primero es el Profitable CEARP para un solo vehículo, y el segundo y el tercero son el Distance-Constrained CEARP y el Min-Max CEARP, respectivamente, ambos para múltiples vehículos. Para resolver estos tres problemas, se han desarrollado métodos exactos y heurísticos que no sólo abordan los problemas previamente definidos en la literatura científica, sino también nuevas generalizaciones que acercan los resultados teóricos a las necesidades reales que pueden surgir. De hecho, la finalidad de la tesis es doble. Por un lado, contribuir a un estudio en profundidad de las variantes definidas del CEARP. Por otro lado, aportar nuevas ideas y conocimientos al campo de la Investigación Operativa que puedan ser útiles para abordar otros problemas combinatorios de naturaleza similar.

En los primeros capítulos se presenta el contexto en el que surgen los problemas estudiados en esta tesis. Empezamos describiendo algunos conceptos esenciales de la Programación Matemática para proporcionar a los lectores no expertos una revisión conceptual detallada y para introducir los temas, la terminología y la notación matemática que se utiliza en los capítulos siguientes. Presentamos varios principios básicos de la teoría de grafos, la programación lineal y entera, y la teoría poliédrica y la combinatoria poliédrica. Posteriormente, ofrecemos una visión general de algunos problemas de rutas clásicos que han sido ampliamente estudiados en la literatura científica. Debido a la gran variedad de problemas de rutas, nos centramos únicamente en aquellos considerados como la base de muchos otros problemas relacionados que han surgido a lo largo de los años debido a la necesidad de adaptarlos a situaciones específicas. El conocimiento de estos problemas originales contribuye a una mejor y más rápida comprensión de los problemas más complejos, ya que para resolver este tipo de problemas es necesario estudiar sus propiedades y características, que son esenciales para desarrollar un método de solución a medida. Para concluir la sección introductoria, se presenta el estado del arte y las aplicaciones del mundo real de los CERP. Con la intención de introducir estos problemas, se incluye una visión general del CETSP, el problema homólogo para problemas de rutas por nodos, junto con un estudio más detallado de toda la investigación existente hasta el momento para el CEARP, el ARP que sirve de base de los problemas estudiados en esta tesis.

A continuación nos centramos en el estudio de la primera generalización del CEARP, el Profitable CEARP (PCEARP). Los problemas de rutas con beneficios tratan situaciones en las que los clientes a los que hay que dar servicio deben ser elegidos entre un conjunto de clientes potenciales que tienen un beneficio asociado, beneficio que se se recoge cuando se les da servicio. Consisten básicamente en diseñar una o varias rutas que den servicio a los clientes elegidos y de forma que se optimice un objetivo definido en función del coste o/y del beneficio. Un problema de rutas con beneficios se denomina *profitable* cuando su objetivo es maximizar la diferencia entre el beneficio recaudado y el coste de las rutas. En el capítulo 5 se estudia en profundidad el PCEARP, en el que se asocia un beneficio a cada cliente y se recolecta (sólo una vez) cuando se sirve al cliente. El objetivo es encontrar un recorrido que maximice la diferencia entre el beneficio total recaudado de servir a los clientes y la distancia de recorrida. El capítulo comienza definiendo formalmente el problema, proponiendo una formulación y realizando un estudio poliédrico en profundidad. Se demuestra que algunas desigualdades de la formulación siempre definen la faceta y que otras lo hacen bajo condiciones específicas. Además, a partir de las propiedades de los recorridos del PCEARP, se presentan otras desigualdades válidas no obtenidas directamente de la formulación, que refuerzan la descripción del poliedro.

Para resolver el Profitable CEARP se ha diseñado e implementado una heurística y un algoritmo Branch and Cut. La heurística combina un procedimiento constructivo y una búsqueda local de manera que somos capaces de proporcionar al algoritmo exacto cotas inferiores iniciales. En el algoritmo Branch and Cut se estudian todos los procedimientos de separación para la identificación de las desigualdades violadas y el orden en que se aplican. Ambos algoritmos han requerido de un ajuste y evaluación mediante diversos experimentos y un amplio análisis estadístico. Para probarlos se han generado cuatro conjuntos diferentes de instancias específicas de este problema con hasta 800 clientes, 400 vértices y 2000 arcos. Según los resultados de los experimentos computacionales, el procedimiento exacto es capaz de resolver óptimamente instancias grandes con hasta 600 clientes, 300 vértices y 1500 arcos, en menos de una hora. El capítulo es el resultado de una colaboración con el profesor Bianchessi, de la Università degli Studi di Milano, destino de mi estancia de investigación. El siguiente trabajo ha sido enviado a una revista internacional para su publicación:

> N. Bianchessi, Á. Corberán, I. Plana, M. Reula, J.M. Sanchis.
> 2021. The Profitable Close Enough Arc Routing Problem.
> Under revision.

Los capítulos 6 y 7 tratan el Distance-Constrained CEARP (DC-CEARP). Se trata de un CEARP en el que una flota de vehículos, o bien un vehículo varias veces, realiza el servicio a los clientes. Esta generalización del problema consiste en encontrar un conjunto de rutas que salgan y entren en el depósito y sirvan a todos los clientes, de forma que la longitud (en distancia o tiempo) de cada ruta no supere un determinado valor. El objetivo es minimizar la longitud total recorrida.

El Capítulo 6 aborda el CEARP con distancias restringidas, definiendo formalmente el problema, introduciendo la notación utilizada y presentando la formulación más prometedora propuesta en la literatura. Dado que en la batería de instancias que existen del problema los valores de las distancias máximas por ruta son muy ajustados y dificultan su resolución, hay un buen número de instancias sin resolver. Por ello, nos centramos en el diseño e implementación de una matheurística multi-arranque que incorpora un método efectivo de Branch and Cut para el CEARP con el fin de optimizar las rutas obtenidas. En los experimentos computacionales, presentamos los resultados obtenidos con dos versiones del algoritmo, considerando un número máximo de iteraciones y un límite de tiempo, respectivamente. En general, el enfoque propuesto encuentra soluciones factibles para casi todas las instancias y soluciones óptimas para más del 70% de ellas. El capítulo se basa en el siguiente documento publicado:

> Á. Corberán, I. Plana, M. Reula, J.M. Sanchis. 2019. A matheuristic for the Distance-Constrained Close Enough Arc Routing Problem. **TOP**. 27, 312–326.

El Capítulo 7 también aborda el Distance Constrained Close Enough Arc Routing Problem, pero en este caso se realiza un estudio más detallado del problema. Comenzamos proponiendo una nueva formulación que combina las mejores características de las formulaciones ya existentes en la literatura ya que, a pesar de tener más variables, se pretende reforzar su relajación lineal. Para esta formulación hemos realizado un estudio exhaustivo de su poliedro asociado y hemos propuesto varias familias de desigualdades válidas. Además, basándonos en los algoritmos de separación para las nuevas desigualdades, hemos propuesto un algoritmo de Branch and Cut que proporciona muy buenos resultados. Se han realizado amplios experimentos computacionales sobre un conjunto de instancias de referencia para analizar la contribución de las desigualdades válidas y los algoritmos de separación presentados. Se comparan los huecos en el nodo raíz y los perfiles de rendimiento de las distintas versiones de nuestro procedimiento de ramificación y corte (utilizando distintas combinaciones de algoritmos de separación). Los resultados de las dos mejores versiones de nuestro

algoritmo Branch and Cut se comparan con los obtenidos con los métodos matheuristicos y exactos de la literatura. La mejor versión de nuestro algoritmo Branch and Cut es capaz de resolver óptimamente instancias con hasta 140 clientes, 196 vértices, 544 arcos y 5 vehículos tomando un tiempo máximo de computación de dos horas. El capítulo se basa en el siguiente documento publicado:

> Á. Corberán, I. Plana, M. Reula, J.M. Sanchis. 2021. On the Distance-Constrained Close Enough Arc Routing Problem. **European Journal of Operational Research.** 291(1), 32-51.

En el Capítulo 8 estudiamos el Min-Max CEARP (MM-CEARP) para una flota de vehículos homogéneos. Teniendo en cuenta el tipo de aplicaciones del CEARP, esta variante trata de equilibrar la longitud de las rutas minimizando la duración de la ruta más larga. El problema consiste en encontrar un conjunto de rutas, todas ellas con inicio y fin en el depósito, que sirvan conjuntamente a todos los clientes, y con un objetivo minmax, es decir de minimizar la máxima ruta. Basándonos en la experiencia previa en el DC-CEARP, los algoritmos Branch and Cut eran muy complicados de resolver cuando intentábamos resolver instancias con muchos vehículos. Por lo tanto, nos planteamos el reto de diseñar e implementar un algoritmo Branch and Price capaz de resolver instancias con un gran número de vehículos. Comenzamos presentando y proporcionando dos modelos diferentes para el problema: una formulación basada en arcos, con variables de flujo y servicio, que se ha utilizado para desarrollar un algoritmo Branch and Cut, y una formulación basada en variables por ruta, que se ha utilizado para el desarrollo de un algoritmo de Branch and Price. También desarrollamos un algoritmo heurístico que utilizamos para proporcionar soluciones iniciales factibles a los algoritmos exactos. El Branch and Cut se basa en en el algoritmo exacto desarrollado en la literatura para el DC-CEARP.

En el Branch and Price hemos implementado dos técnicas atípicas de estos algoritmos a la hora de abordar problemas de rutas: la regla de primer nivel aplicada en el esquema de ramificación (basada en la regla de ramificación de Ryan y Foster) y un algoritmo de Branch and Cut para resolver óptimamente los Pricing Prlblems en lugar de utilizar programación dinámica. El primero permite recuperar soluciones enteras a costa de una diversificación de los conjuntos de rutas factibles, no introduce simetrías en el espacio de soluciones, no altera la estructura de los problemas de precios y, por último, permite que los pricing problems sigan compartiendo la misma región factible. Hemos realizado un amplio análisis computacional en el que hemos comparado el rendimiento de los algoritmos exactos en cuatro conjuntos de instancias probadas con un mínimo de 2 vehículos y un máximo de 11 vehículos. Los resultados muestran que

el algoritmo de Branch and Cut consigue los mejores resultados para las instancias con dos vehículos, mientras que el algoritmo Branch and Price se comporta mejor para las instancias con tres o más vehículos. Este capítulo también es resultado de la colaboración con el profesor Bianchessi, de la Università degli Studi di Milano, durante mi estancia de investigación. El trabajo ha sido enviado a una revista internacional para su publicación:

> N. Bianchessi, Á. Corberán, I. Plana, M. Reula, J.M. Sanchis. 2021. The Min-Max Distance-Constrained Close Enough Arc Routing Problem. Under revision.

Aunque la tesis concluye aquí, la investigación es un campo interminable por lo que el estudio llevado a cabo en este trabajo puede continuar abordando otras restricciones y variantes de los problemas descritos. Además, muchas de las ideas en las que se basan algunos de los algoritmos diseñados podrían ser adaptados directa o fácilmente para abordar otros problemas de enrutamiento de arcos relacionados, así como muchas de las ideas presentadas en las partes teóricas también podrían ser utilizadas en otros problemas relacionados.

Dados los excelentes resultados del algoritmo Branch and Price en la resolución de instancias de Min-Max CEARP con un número de vehículos grande, valdría la pena desarrollar un algoritmo del mismo tipo para el Distance-Constrained CEARP. Se ha estudiado en la literatura que los algoritmos Branch and Cut son muy eficientes en la solución del CEARP, y aquí mostramos que son muy eficientes en la solución de la variante multi-vehículo cuando el tamaño de la flota es pequeño pero, cuando el número de vehículos aumenta, un algoritmo de Branch and Price es mucho más apropiado.

Con respecto al Profitable CEARP considerado en el Capítulo 5, podría ampliarse de varias maneras añadiendo progresivamente características de problemas reales relacionados. Por ejemplo, el beneficio por cliente podría definirse por el tiempo que se tarda en realizar el servicio. Además, si un solo vehículo no es capaz de realizar todos los servicios, se requiere una flota de vehículos (o múltiples rutas para un solo vehículo). Se podría considerar que todos los vehículos tienen las mismas características o que tenemos una flota heterogénea. También podríamos establecer una distancia/tiempo máximo para cada ruta o minimizar el coste de la ruta más larga.

A excepción de la matheurística propuesta para el DC-CEARP, a lo largo de esta tesis hemos abordado los problemas desde un punto de vista teórico, cuyos resultados se han utilizado para desarrollar algoritmos exactos para la solución óptima de los

problemas. Todos los problemas aquí estudiados son problemas combinatorios que son NP-*hard*, por lo que no existen algoritmos que encuentren las soluciones óptimas en tiempo polinómico. Así, puede ocurrir que en determinadas circunstancias sea necesario disponer de soluciones de buena calidad para problemas reales en tiempos de computación cortos. En este sentido, pensamos que, como resultado del conocimiento adquirido durante el estudio de los problemas tratados en esta tesis, se podrían diseñar e implementar algoritmos heurísticos y/o metaheurísticos para los tres problemas estudiados que mejoren la calidad de las soluciones.

# Bibliography

Applegate, D., Cook, W., Dash, S., Rohe., A., 2002. Solution of a min-max vehicle routing problem. INFORMS Journal on Computing 14, 132–143.

Aráoz, J., Fernández, E., Franquesa, C., 2017. The generalized arc routing problem. TOP 25, 497–525.

Aráoz, J., Fernández, E., Meza, O., 2009. Solving the prize-collecting rural postman problem. European Journal of Operational Research 196, 886–896.

Aráoz, J., Fernández, E., Zoltan, C., 2006. Privatized rural postman problems. Computers & Operations Research 33, 3432–3449.

Archetti, C., Guastaroba, G., Speranza, M., 2014a. An ILP-refined tabu search for the directed profitable rural postman problem. Discrete Applied Mathematics 163, 3–16.

Archetti, C., Speranza, M., 2014. Arc routing problems with profits, in: Corberán, Á., Laporte, G. (Eds.), Arc Routing: Problems, Methods, and Applications. SIAM. MOS-SIAM Series on Optimization, pp. 281–299.

Archetti, C., Speranza, M., Vigo, D., 2014b. Vehicle routing problems with profits, in: Toth, P., Vigo, D. (Eds.), Vehicle Routing: Problems, Methods, and Applications. SIAM. MOS-SIAM Series on Optimization, pp. 273–297.

Arkin, E., Hassin, R., 1994. Approximation algorithms for the geometric covering salesman problem. Discrete Applied Mathematics 55, 197–218.

Assad, A., Ball, M., Bodin, L., Golden, B., 1983. Routing and scheduling of vehicles and crews. Computers & Operations Research 10, 63–211.

Assad, A., Golden, B., 2000. Arc Routing Methods and Applications. Handbooks of Operations Research and Management Science 8, North Holland.

Ávila, T., Corberán, Á., Plana, I., Sanchis, J., 2015. The stacker crane problem and the directed general routing problem. Networks 65, 43–55.

Ávila, T., Corberán, Á., Plana, I., Sanchis, J., 2016a. A branch-and-cut algorithm for the profitable windy rural postman problem. European Journal of Operational Research 249, 1092 – 1101.

Ávila, T., Corberán, Á., Plana, I., Sanchis, J.M., 2016b. A new branch-and-cut algorithm for the generalized directed rural postman problem. Transportation Science 50, 750–761.

Ávila, T., Corberán, Á., Plana, I., Sanchis, J.M., 2017. Formulations and exact algorithms for the distance-constrained generalized directed rural postman problem. EURO Journal on Computational Optimization 5, 339–365.

Bachem, A., Grötschel, M., 1982. New aspects of polyhedral theory. Modern Applied Mathematics: Optimization and Operations Research (B. Korte, eds). North Holland, Amsterdam , 51–106.

Baldacci, R., Boschetti, M.A., Maniezzo, V., Zamboni, M., 2005. Scatter Search Methods for the Covering Tour Problem. Springer US, Boston, MA. pp. 59–91.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H., 1998. Branch-and-price: column generation for solving huge integer programs. Operations Research 46, 316–329.

Bazaraa, M., Jarvis, J., 1977. Linear Programming and Network Flows. Wiley, New York.

Behdani, B., Smith, J., 2014. An integer-programming-based approach to the close-enough traveling salesman problem. INFORMS J Comput 26, 415 – 432.

Beltrami, E.J., Bodin, L.D., 1974. Networks and vehicle routing for municipal waste collection. Networks 4, 65–94.

Benavent, E., Campos, V., Corberán, Á., Mota, E., 1983. Problemas de rutas por arcos. Qüestió 7, 479–490.

Benavent, E., Corberán, Á., Gouveia, L., Mourão, M.C., Pinto, L., 2015. Profitable mixed capacitated arc routing and related problems. TOP 23, 244 – 274.

Benavent, E., Corberán, Á., Plana, I., Sanchis, J., 2014. Arc routing problems with min-max objetives, in: Corberán, Á., Laporte, G. (Eds.), Arc Routing: Problems, Methods and Applications. MOS-SIAM Series on Optimization, Philadelphia, pp. 255–280.

Berge, C., 1973. Graphs and Hypergraphs. North Holland, Amsterdam.

Bianchessi, N., Corberán, Á., Plana, I., Reula, M., Sanchis, J.M., 2021. The profitable close enough arc routing problem. Submitted .

Bodin, L., Golden, B., 1981. Classification in vehicle routing and scheduling. Networks 11, 97–108.

Bondy, L., Murty, U., 1976. Graph Theory with Applications. American Elsevier, New York.

Carrabs, F., Cerrone, C., Cerulli, R., Gaudioso, M., 2017. A novel discretization scheme for the close enough traveling salesman problem. Computers & Operations Research 78, 163 – 171.

Cerrone, C., Cerulli, R., Golden, B., Pentangelo, R., 2017. A flow formulation for the close-enough arc routing problem, in: Sforza, A., Sterle, C. (Eds.), Optimization and Decision Science: Methodologies and Applications. ODS 2017. Springer. volume 217 of *Springer Proceedings in Mathematics & Statistics*, pp. 539–546.

Christofides, N., 1975. Graph Theory. An Algorithmic Approach. Academic Press, New York.

Chvátal, V., 1983. Linear Programming. Freeman, New York.

Clarke, G., Wright, J.W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations research 12, 568–581.

Colombi, M., Mansini, R., 2014. New results for the directed profitable rural postman problem. European Journal of Operational Research 238, 760 – 773.

Corberán, Á., Eglese, R., Hasle, G., Plana, I., Sanchis, J., 2021. Arc routing problems: A review of the past, present, and future. Networks 77, 88–115.

Corberán, Á., Laporte, G., 2014. Arc Routing: Problems, Methods, and Applications. MOS-SIAM Series on Optimization, Philadelphia.

Corberán, Á., Letchford, A., Sanchis, J., 2001. A cutting plane algorithm for the general routing problem. Mathematical Programming 90, 291–316.

Corberán, Á., Plana, I., Reula, M., Sanchis, J., 2019. A matheuristic for the distance-constrained close-enough arc routing problem. TOP 27, 312–326.

Corberán, Á., Plana, I., Reula, M., Sanchis, J.M., 2021. On the distance-constrained close enough arc routing problem. European Journal of Operational Research 291, 32–51.

Corberán, Á., Plana, I., Rodríguez-Chía, A., Sanchis, J., 2013. A branch-and-cut for the maximum benefit chinese postman problem. Mathematical Programming 141, 21–48.

Corberán, Á., Prins, C., 2010. Recent results on arc routing problems: An annotated bibliography. Networks 56, 50–69.

Corberán, Á., Romero, A., Sanchis, J., 2003. The mixed general routing problem polyhedron. Mathematical Programming 96, 103–137.

Corberán, Á., Sanchis, J., 1994. A polyhedral approach to the rural postman problem. European Journal of Operational Research 79, 95–114.

Cornuèjols, G., Fonlupt, J., Naddef, D., 1985. The traveling salesman problem on a graph and some related inequalities. Mathematical Programming 33, 1–27.

Coutinho, W., Subramanian, A., do Nascimento, R., Pessoa, A., 2016. A branch-and-bound algorithm for the close enough traveling salesman problem. INFORMS J Comput 28, 752 – 765.

Dantzig, G., 1963. Linear Programming. Freeman, New York.

Dantzig, G., Ramser, J.H., 1959. The truck dispatching problem. Management Science 6, 80–91.

Dantzig, G., Wolfe, P., 1960. Decomposition principle for linear programs. Operations Research 8, 101–111.

Desaulniers, G., Desrosiers, J., Solomon, M. (Eds.), 2005. Column Generation. Springer, New York.

Dolan, E., More, J., 2002. Benchmarking optimization software with performance profiles. Mathematical Programming 91, 201–213.

Dong, J., Yang, N., Chen, M., 2007. Heuristic approaches for a TSP variant: The automatic meter reading shortest tour problem, in: Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies. Springer, pp. 145–163.

Drexl, M., 2007. On some generalized routing problems. Ph.D. thesis. Rheinisch-Westfälische Technische Hochschule, Aachen University.

Drexl, M., 2014. On the generalized directed rural postman problem. Journal of the Operational Research Society 65, 1143–1154.

Dror, M., 2000. Arc Routing: Theory, Solutions and Applications. Kluwer Academic Publishers.

Dumitrescu, A., Mitchell, J., 2003. Approximation algorithms for TSP with neighborhoods in the plane. Journal of Algorithms 48, 135–159.

Edmonds, J., 1965. The Chinese postman problem. Operations Research 13 Supplement 1, B73.

Edmonds, J., Johnson, E., 1973. Matching, Euler tours and the Chinese postman. Mathematical Programming 5, 88–124.

Eglese, R., Golden, B., Wasil, E., 2014. Route optimization for meter reading and salt spreading, in: Corberán, Á., Laporte, G. (Eds.), Arc Routing: Problems, Methods and Applications. MOS-SIAM Series on Optimization, Philadelphia, pp. 303–320.

Eiselt, H., Gendreau, M., Laporte, G., 1995a. Arc routing problems. part I: The Chinese postman problem. Operations Research 43, 231–242.

Eiselt, H., Gendreau, M., Laporte, G., 1995b. Arc routing problems. part II: The rural postman problem. Operations Research 43, 399–414.

Feillet, D., Dejax, P., Gendreau, M., 2005. Traveling salesman problems with profits. Transportation Science 39, 188–205.

Fischetti, M., Salazar-González, J.J., Toth, P., 1995. Experiments with a multicommodity formulation for the symmetric capacitated vehicle routing problem, in: Proceedings of the 3rd meeting of the EURO working group on transportation, pp. 169–173.

Fleischmann, B., 1985. A cutting plane procedure for the traveling salesman problem on road networks. European Journal of Operational Research 21, 147–172.

Fleischmann, B., 1988. A new class of cutting planes for the symmetric traveling salesman problem. Mathematical Programming 40, 225–246.

Frederickson, G., 1979. Approximation algorithms for some postman problems. Journal of the Association for Computing Machinery 26, 538–554.

Frederickson, G., Hecht, M., Kim, C., 1978. Aproximation algorithms for some routing problems. Journal on Computing 7, 178–193.

Garfinkel, R., Nemhauser, G., 1972. Integer Programming. Wiley, New York.

Gendreau, M., Laporte, G., Semet, F., 1997. The covering tour problem. Operations Research 45, 568–576.

Golden, B.L., Wong, R.T., 1981. Capacitated arc routing problems. Networks 11, 305–315.

Grötschel, M., Padberg, M., 1985. Polyhedral theory, in: Lawler, E., Lenstra, J., Rinnooy-Kan, A., Shmoys, D. (Eds.), The Traveling Salesman Problem. J. Wiley & Sons, pp. 251–305.

Guan, M., 1962. Graphic programming using odd and even points. Chinese Mathematics 1, 273–277.

Guan, M., 1984. On the windy postman problem. Discrete Applied Mathematics 9, 41–46.

Gulczynski, D., Heath, J., Price, C., 2006. The close enough traveling salesman problem: A discussion of several heuristics, in: Perspectives in Operations Research. Springer. volume 36 of *Operations Research/Computer Science Interfaces Series*, pp. 217–283.

Hà, M.H., Bostel, N., Langevin, A., Rousseau, L.M., 2012. An exact algorithm for close enough traveling salesman problem, in: Proceedings of the 1st International Conference on Operations Research and Enterprise Systems, pp. 233–238.

Hà, M.H., Bostel, N., Langevin, A., Rousseau, L.M., 2013. An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. European Journal of Operational Research 226, 211–220.

Hà, M.H., Bostel, N., Langevin, A., Rousseau, L.M., 2014. Solving the close enough arc routing problem. Networks 63, 107–118.

Harary, F., 1969. Graph Theory. Addison Wesley, Reading, MA.

Hoffman, K., Padberg, M., 1985. Lp-based combinatorial problem solving. Annals of Operations Research 4, 145–194.

Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., Vanderbeck, F., 2010. Column generation based primal heuristics. Electronic Notes in Discrete Mathematics 36, 695–702.

Jozefowiez, N., 2014. A branch-and-price algorithm for the multivehicle covering tour problem. Networks 64, 160–168.

Karp, R.M., 1972. Reducibility among combinatorial problems. Springer US, Boston, MA. pp. 85–103.

Khachian, L., 1979. A polynomial algorithm for linear programming. Soviet Math. Doklady 20, 191–194.

Lenstra, J., Rinnooy-Kan, A., 1976. On the general routing problem. Networks 6, 273–280.

Lenstra, J., Rinnooy-Kan, A., 1981. Complexity of vehicle routing and scheduling problems. Networks 11, 221–227.

Letchford, A., 1997. New inequalities for the general routing problem. European Journal of Operational Research 96, 317–322.

Malandraki, C., Daskin, M., 1993. The maximum benefit chinese postman problem and the maximum benefit traveling salesman problem. European Journal of Operational Research 65, 218–234.

Mata, C.S., Mitchell, J.S., 1995. Approximation algorithms for geometric tour and network design problems, in: Proceedings of the eleventh annual symposium on Computational geometry, pp. 360–369.

Mennell, W., 2009. Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem. Ph.D. thesis. University of Maryland, College Park.

Minieka, E., 1979. The Chinese postman problem for mixed networks. Management Science 25, 643–648.

Mourão, M.C., Pinto, L.S., 2017. An updated annotated bibliography on arc routing problems. Networks 70, 144–194.

Nemhauser, G., Wolsey, L., 1988. Integer and Combinatorial Optimization. Wiley, New York.

Nobert, Y., Picard, J., 1996. An optimal algorithm for the mixed Chinese postman problem. Networks 27, 95–108.

Orloff, C., 1974. A fundamental problem in vehicle routing. Networks 4, 35–64.

Papadimitriou, C., 1976. On the complexity of edge traversing. Journal of the ACM 23, 544–554.

Pearn, W.L., Chiu, W.C., 2005. Approximate solutions for the maximum benefit Chinese postman problem. International Journal of Systems Science 36, 815–822.

Pearn, W.L., Wang, K., 2003. On the maximum benefit Chinese postman problem. Omega 31, 269–273.

Pecin, D., Uchoa, E., 2019. Comparative analysis of capacitated arc routing formulations for designing a new branch-cut-and-price algorithm. Transportation Science 53, 1673–1694.

Pulleyblank, W., 1983. Polyhedral Combinatorics. Springer Berlin Heidelberg.

Renaud, A., Absi, N., Feillet, D., 2017. The stochastic close-enough arc routing problem. Networks 69, 205–221.

Russo, D.D., Cerrone, C., Di Placido, A., 2019. The mixed-constrained routing problem–a combination of cearp and cetsp. WARP3 Proceedings, Pizzo, Italy .

Ryan, D., Foster, B., 1981. An integer programming approach to scheduling. Computer Scheduling of Public Transport 1, 269–280.

Schrijver, A., 1986. Theory of Linear and Integer Programming. Wiley, Chichester.

Shuttleworth, R., Golden, B., Smith, S., Wasil, E., 2008. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network, in: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Vehicle Routing Problem: Lastest Advances and New Challenges. Springer, pp. 487–501.

Uribe-Pérez, N., Hernández, L., De la Vega, D., Angulo, I., 2016. State of the art and trends review of smart metering in electricity grids. Applied Sciences 6, 1–24.

Vansteenwegen, P., Souffriau, W., Oudheusden, D.V., 2011. The orienteering problem: A survey. European Journal of Operational Research 209, 1–10.

Win, Z., 1987. Contributions to routing problems. Ph.D. thesis. Universität Augsburg.

Yuan, B., Orlowska, M., Sadiq, S., 2007. On the optimal robot routing problem in wireless sensor networks. IEEE Transactions on Knowledge and Data Engineering 19, 1252–1261.